NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

APPLICATIONS AND BENEFITS FOR BIG DATA SETS USING TREE DISTANCES AND THE T-SNE ALGORITHM

by

Suyoung Lee

March 2016

Thesis Advisor: Samuel E. Buttrey
Second Reader: Lyn R. Whitaker

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704–0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2016	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE APPLICATIONS AND BENEFITS FOR BIG DATA SETS USING TREE DISTANCES AND THE T-SNE ALGORITHM			**5. FUNDING NUMBERS**
6. AUTHOR(S) Suyoung Lee			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Modern data sets often consist of unstructured data and mixed data; that is, they include both numerical and categorical variables. Often, these data sets will include noise, redundancy, missing values and outliers. Clustering is one of the most important and widely-used data analytic methods. However, clustering requires the ability to measure distances or dissimilarities, which are not defined in an obvious way for mixed data. Practitioners often use the Gower dissimilarity for this task. In this work we use tree distance computed using Buttrey's treeClust package in R, as discussed by Buttrey and Whitaker in 2015, to process mixed data, at the same time handling missing values and outliers. Visualization is also an important method for big data. We use the t-distributed Stochastic Neighbor Embedded (t-SNE) algorithm for visualization introduced by van der Maaten and Hinton in 2008, which produces visualization for high-dimensional data by assigning individual data points in a two- or three-dimensional map. We also use popular visualization techniques grouped under the name "multidimensional scaling." We compare the results using the tree distance and the t-SNE algorithm to results from using Gower dissimilarity and multidimensional scaling. Unlike established dimensionality reduction techniques, which generally map from high dimensions directly to two (or three) dimensions, we explore a new approach in which the dimensionality reduction takes place in several separate steps. Our experiments show that our new techniques can outperform the established techniques in producing visualizations of high-dimensional mixed data.

14. SUBJECT TERMS big data sets, tree distance algorithm, treeClust, visualization, t-SNE algorithm, dimensionality reduction			**15. NUMBER OF PAGES** 81
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

APPLICATIONS AND BENEFITS FOR BIG DATA SETS USING TREE DISTANCES AND THE T-SNE ALGORITHM

Suyoung Lee
Captain, Republic of Korea Army
B.Sc., Korea Maritime and Ocean University, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 2016

Approved by: Samuel E. Buttrey
Thesis Advisor

Lyn R. Whitaker
Second Reader

Patricia A. Jacobs
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Modern data sets often consist of unstructured data and mixed data; that is, they include both numerical and categorical variables. Often, these data sets will include noise, redundancy, missing values and outliers. Clustering is one of the most important and widely-used data analytic methods. However, clustering requires the ability to measure distances or dissimilarities, which are not defined in an obvious way for mixed data. Practitioners often use the Gower dissimilarity for this task. In this work we use tree distance computed using Buttrey's treeClust package in R, as discussed by Buttrey and Whitaker in 2015, to process mixed data, at the same time handling missing values and outliers. Visualization is also an important method for big data. We use the t-distributed Stochastic Neighbor Embedded (t-SNE) algorithm for visualization introduced by van der Maaten and Hinton in 2008, which produces visualization for high-dimensional data by assigning individual data points in a two- or three-dimensional map. We also use popular visualization techniques grouped under the name "multidimensional scaling." We compare the results using the tree distance and the t-SNE algorithm to results from using Gower dissimilarity and multidimensional scaling. Unlike established dimensionality reduction techniques, which generally map from high dimensions directly to two (or three) dimensions, we explore a new approach in which the dimensionality reduction takes place in several separate steps. Our experiments show that our new techniques can outperform the established techniques in producing visualizations of high-dimensional mixed data.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CMDS	Classical Multidimensional Scaling
MDS	Multidimensional Scaling
PCA	Principal Component Analysis
SNE	Stochastic Neighbor Embedded
t-SNE	t-distributed Stochastic Neighbor Embedded

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Most big data sets consist of unstructured data and mixed data, that is they contain both numerical and categorical variables. In data analytics, clustering is one of the most important methods for obtaining valuable information. Many clustering approaches require a measure of distance between observations. One such measure is the tree distance, which measures proximity between observations of mixed-type data while handling missing values and outliers. We use the treeClust package of Buttrey (2015) in R data analysis software, discussed by Buttrey and Whitaker (2015b), to compute these distances.

Visualization is also an important method for big data analytics. We use the t-distributed Stochastic Neighbor Embedded (t-SNE) algorithm for visualization introduced by van der Maaten and Hinton (2008). The t-SNE algorithm produces visualizations of high-dimensional data by assigning individual data points in a two or three-dimensional map (van der Maaten & Hinton, 2008). It is especially effective for high-dimensional data that consists of a large number of classes and produces more discernible visualizations than other techniques. We also use classical multidimensional scaling (CMDS), which is one of the most popular visualization techniques.

In this thesis, we compare the tree distance algorithm to the most popular measure of inter-point distance, which is the Gower dissimilarity, and compare the t-SNE algorithm to other visualization technique, including CMDS and two non-metric competitors. We also explore a dimensionality reduction technique using the t-SNE algorithm. Unlike established dimensionality reduction techniques, which reduce the dimensionality from the original high number of dimensions to two or three dimensions directly, we apply dimensionality reduction in a "long path" reducing the dimension gradually (e.g., from the original dimensionality to 100 to 60 to 30 to 2). We operate on several well-

known data sets and compare the performance of the two distance measures and the different scaling techniques.

This thesis concludes with some issues concerning dimensionality reduction. When we try to conduct long-path dimensionality reduction, it sometimes gives more discernible visualization than the established techniques. However, the algorithm also produces errors of unknown cause. Finally, we suggest more research into the long path dimensionality reduction technique for more discernible visualization.

References

Buttrey, S. E., & Whitaker, L. R. (2015a). A scale-independent, noise-resistant dissimilarity for tree-based clustering of mixed data (submitted), Naval Postgraduate School, Monterey, CA.

Buttrey, S. E., & Whitaker, L. R. (2015b). treeClust: An R package for tree-based clustering dissimilarities. *The R Journal*, *7*(2), 227–236.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *1* (2008) 1–48

ACKNOWLEDGMENTS

I would like to thank Professor Buttrey and Professor Whitaker for allowing me to work with them. It is an honor and I really enjoyed working with them.

I also would like to thank my government for giving me the chance to have such a valuable experience at the Naval Postgraduate School.

Lastly, I would like to thank my family, who always encourages and supports me.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

"Big data" is a broad term for extremely large and complex sets of data for which traditional applications are inappropriate (Oguntimilehin & Ademola, 2014). More specifically, big data refers to datasets that cannot be "acquired, managed, and processed" by established technologies and tools within a reasonable time (Chen, Mao, Zhang, & Leung, 2014). Big data analytics is the process of analyzing large data sets having numerical, categorical, and other variables to discover "hidden patterns, unknown correlations, market trends, customer preferences and other useful information" (Rouse, 2014). The results can improve the efficiency of operations, and increase profits, quality of customer service, and effectiveness in marketing. Specifically, the government can achieve cost benefits, improvement in productivity, and innovation utilizing big data within public institutions. Big data analytics can also be applied to military problems. Further, large numbers of data sets in the military, such as manpower data, are both big and of mixed data types, having both numerical and categorical variables.

A number of researchers have developed techniques to analyze and find patterns in mixed-type multidimensional data over the years. One important consideration in these data sets is defining a suitable measure of inter-point distance. One such measure is the widely-used dissimilarity of Gower (1971). Buttrey and Whitaker (2015a) implemented and expanded the competing "tree distance" algorithm in the R data analysis software (R Core Team, 2015), through the package "treeClust" (Buttrey, 2015). This technique seems to hold advantages for high-dimensional and mixed-type data sets. Once an inter-point distance is defined, it can be useful to map the high-dimensional data into low dimensions for the purposes of visualization and interpretation. Among the techniques for mapping data to lower dimensions, the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm (van der Maaten & Hinton, 2008) is well

suited for high-dimensional data. In particular, the t-SNE algorithm produces high-quality visualizations by minimizing the tendency of points mapped from very high dimensions to gather at the center of the low dimensional map. Although t-SNE was originally developed to visualize numeric data in Euclidean space, in combination with a measure of dissimilarity for mixed-type data, such as those produced by the tree distance algorithm, t-SNE can be used to visualize high-dimensional mixed-type data.

In this thesis, we compare the results of visualizations of high-dimensional data by using the tree distance algorithm together with the t-SNE algorithm to the results produced by other dissimilarity measures like that of Gower and other mapping techniques like classical multidimensional scaling (CMDS), to examine the ability of tree distances and t-SNE to improve visualization.

B. OBJECTIVES AND THESIS OUTLINE

In this thesis, we describe a measure of inter-point distance, the tree distance produced by the treeClust algorithm that we use for our study, and a visualization technique, which is the t-SNE algorithm suitable for our study. We use t-SNE as a tool for dimensionality reduction. Unlike the usual dimensionality reduction technologies, which map from high-dimensional space to two or three dimensions directly, we apply dimensionality reduction several times reducing the dimension each time (e.g., from dimension 100 to 60 to 30 to 2). We use a number of data sets from the literature to compare the results from conducting a sequence of dimensionality reductions to one-time dimensionality reduction using t-SNE and also using other multidimensional scaling techniques.

We discuss how well our new, “long path,” technique performs in terms of the goals of dimensionality reduction: to maintain as much as possible of the structure in high dimensions in the two- or three-dimensional visualization (van der Maaten & Hinton, 2008). We finish the thesis by suggesting the long path dimensionality reduction technique for visualization.

The outline of the thesis is as follows. In Chapter II, we introduce the tree distance algorithm as presented by Buttrey and Whitaker (2015a; 2015b), which we use to produce inter-point distances, and introduce the t-SNE algorithm as presented by van der Maaten and Hinton (2008), which we use as a visualization technique. In Chapter III, we present the methodology and experimental setup we used to evaluate dimensionality reduction. In Chapter IV, we present and discuss the results produced from our experiments. In Chapter V, conclusions and recommendations for future work are presented.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LITERATURE REVIEW

In this chapter, we review overall aspects of big data analysis today and examine two major methods for clustering and dimensionality reduction mapping: the use of the tree distance algorithm implemented and expanded by Buttrey and Whitaker (2015a; 2015b) and the t-SNE algorithm introduced by van der Maaten and Hinton (2008). We also describe the data sets used to explore our new technique in this chapter.

The chapter is organized as follows: Section A introduces the overall aspects of big data analysis today. Section B describes clustering and the importance of clustering. In Section C, we describe the tree distance algorithm and its benefits when it is used for analysis of big data. Section D describes the t-SNE algorithm. In Section E, we demonstrate dimensionality reduction for visualizing the data. Finally, Section F introduces the data sets we use for our experiment.

A. BIG DATA AND BIG DATA SETS

In this section, we demonstrate some of the properties of big data. Big data is a "large volume of data—both structured and unstructured—that inundates a business on a day-to-day basis" (SAS, n.d.).

Laney (2001) characterized big data by "three v's": volume, velocity, and variety. In these paragraphs we examine these three facets of big data.

Volume: There has been exponential growth in the size of data size in recent years. According to IBM, ninety percent of the all data today was created in the past 2 or 3 years and every day we create 2.5 quintillion bytes of data. A study by the Institute for Digital Communications predicts that we will have 50 times that amount of data by 2020 (Mearian, 2011). Storing big data was a problem in the past, but this is becoming easier owing to the development of new technologies (e.g., Hadoop [The Apache Software Foundation, 2014]).

Velocity: In the past it could take considerable time for computers and servers to acquire and process data. But now with the advent of the World Wide Web, data is often created in real time and computers and devices are expected to process the data immediately.

Variety: most of the data in the past was structured data, for example, in the form of numeric matrices. Today the data becomes more unstructured, complicated—having both numeric and categorical data. Data can be stored in various formats: structured, semi-structured, unstructured, and mixed data (e.g., text documents, email, video, and audio).

Marr (2015) explains two more V's that describe the properties of big data today more completely: veracity and value.

Veracity: This represents confidence in the data. Big data's quality and accuracy are hard to control, with problems like hashtags, abbreviations, and typographical errors. Now it is becoming possible to deal with these data types through the technology of big data analytics.

Value: The final V represents the capability to turn the data into value. Big data can deliver value in almost any area of business or society. Value is an important issue in a big data society today and many data scientists are trying to develop ways to get valuable information from big data.

Big data is not just a large quantity of data; it is also a concept that allows us to understand the existing data in new ways and helps us interpret existing data and analyze future data (Pinal, 2013). Big data analytics is an important progress in big data practices, and if utilized effectively, it can produce a lot of benefits to the field (Burbank, 2016).

B. CLUSTERING

Clustering is another important skill in big data analytics by which to extract valuable information. It is the process of organizing data into groups according to certain properties or similarities. Clustering is used to discover

natural groups or underlying structure of a given data set in, for example, text mining, social network analysis, bioinformatics, market research, and many other fields (Hu & Kaabouch, 2013). That is, clusters are sets of data points that share similar attributes, and clustering algorithms are the techniques that group these data points into different clusters based on their similarities.

The significant part of most clustering algorithms is the measurement of proximity between two observations. The proximity is a measurement of the similarity or dissimilarity. If the larger proximity value for two objects means that they are close, the measurement can be referred to as similarity. If the larger proximity value for two objects means that they are very different, it can be referred to as dissimilarity. The proximities can be of different data types and can be measured on different data scales—binary, discrete, continuous, qualitative and quantitative.

Buttrey and Whitaker (2015a) assert that a dissimilarity measure can be expected to have certain qualities. First, it should incorporate both numerical and categorical variables. Second, it should be insensitive to linear scaling of numeric variables. Third, it should permit the incorporation of variable-specific weights so that some variables can be made more influential than others. Fourth, it should detect the common situation where two variables contain identical information and prevent those variables from being double-counted. That is, it should be able to adjust for correlation among variables. Fifth, it should be insensitive to extreme outliers in the data. Sixth, it should operate in the presence of missing data. Seventh, it should be straightforward to compute, even in large data sets.

To satisfy these qualities, many data scientists are working on developing new technologies, but there are still unsolved problems with measuring proximities and hence with clustering. Processing large amounts of complex data can be a problem, because computation time can be intolerable (Eynard, 2009). In addition, differing clustering results can be produced by different algorithms.

C. TREE DISTANCE ALGORITHM

In this thesis, we describe the tree distance algorithm, which is well suited for computing inter-point distances in big data sets, and we use its implementation in the treeClust package in R.

Tree distances have several advantages for measuring dissimilarities among observations (Buttrey & Whitaker, 2015a). First, the tree distance algorithm works on mixed data sets, which have both numeric and categorical variables. The algorithm builds one tree per variable, treating each variable, in turn, as the response and the remaining variables as predictors. For numeric responses, regression trees are built and for categorical responses, classification trees are built. Second, the distance is resistant to noise variables and unlike Gower dissimilarities (Gower, 1966), tree distances are resistant to outliers. Third, the tree-distance algorithm is invariant to different scales of the data and resistant to monotonic functions of the variables.

The central idea of the tree distance algorithm is that two observations are similar if they tend to fall in the same leaves of classification or regression trees (Buttrey & Whitaker, 2015a). For a data set with p variables, the algorithm creates p trees, each variable serving as the response variable for one tree, with the others acting as predictors. It also uses cross-validation to prune each tree to an optimal size and selects the size for which the cross-validated error rate is minimized. The treeClust package in R implements this algorithm.

A tree built with a noise variable as the response often has a pruned size of 1 and classifies every observation into the same leaf, so it contributes nothing to the dissimilarity computations. Let the label of the leaf of the t^{th} tree into which the i^{th} observation falls be denoted by $L_t(i)$. Then the algorithm measures the dissimilarity between observations i and j by

$$d(i,j) = \sum_{t=1}^{p} \begin{cases} 0 & \text{if } L_t(i) = L_t(j) \\ d^t(i,j) & \text{if } L_t(i) \neq L_t(j) \end{cases},$$

where $d^t(i,j)$ is the "inter-leaf" distance, which is the distance between leaf i and j for tree t.

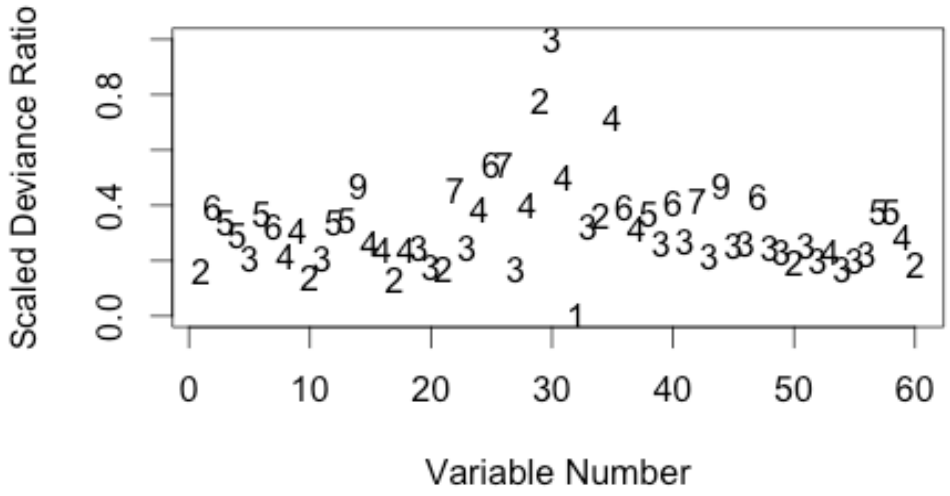
The package supplies four options for the specific form of d^t. For the distance called d_1 for example, $d^t(i,j) = 1$ when $L_t(i) \neq L_t(j)$. After a tree is built, the algorithm computes the sum of deviances in its leaves. A tree's quality can be measured by the ratio of the change in deviance between root and leaves to the deviance at the root. This ratio is denoted by q, a number between zero and 1. A tree with a large q is presumably better able to help cluster individual observations. For the distance d_2, therefore, each tree gets a weight based on how big its q is compared to the largest q observed across all trees. That means when observations i and j fall in the same leaf of tree t, then $d_2^t(i,j) = 0$, and otherwise is $q_t / \max_k(q_k)$. A third distance, d_3, accounts for distances among the leaves within a specific tree, and a fourth, d_4, uses d_3 but also assign weights to trees as d_2 does. Buttrey and Whitaker (2015a, pp. 5–6) show a hypothetical example of a tree in their paper.

The treeClust package includes options for clustering and measures the clustering solution's quality by Cramer's V (Cramér, 1999), which is the usual χ^2 measure of association for the two-way table, scaled to produce a number between 0 and 1. Cramer's V will be small when the cluster labels assigned by the clustering algorithm do not follow class labels representing actual cluster membership well, and close to 1 when most clusters correspond to classes.

Figure 1 is a picture of the treeClust output for the "splice" data (see section F). This picture shows the deviance ratio on the y axis, scaled to have maximum 1, and the tree number (or the corresponding variable number) on the x axis. Each point shown by a digit gives the size (the number of leaves) of a tree. The splice data has 60 variables, which means the treeClust algorithm makes 60 trees. After pruning only 59 trees are left. We can see the number "1" at x=32. The number "1" means that the tree for the 32nd variable was pruned

down to the root node and dropped from the distance computation. The best tree—the one whose deviance ratio is highest—is number 30; that tree has three leaves.

Figure 1. The treeClust plot for the splice data



Another distance we use in this work is the Gower distance. This well-known distance is especially suited for handling mixed-type data. Gower (1971) introduced the distance between i and j across variables p as the average of all component-wise distances. The Gower distance is defined as

$$S_{ij} = \sum_{k=1}^{p} \delta_{ijk} S_{ijk} \Big/ \sum_{k=1}^{p} \delta_{ijk}$$, where

S_{ijk} is a dissimilarity score for x_i and x_j on variable k, $k = 1,\ldots, p$, that ranges between 0 and 1. For a numeric variable, S_{ijk} is defined as $|x_{ik} - x_{jk}| / (\max_l(x_{lk}) - \min_l(x_{lk}))$. For categorical variables, S_{ijk} is 0 if $x_{ik} = x_{jk}$ and otherwise 1. δ_{ijk} adjusts for the ability to make comparisons, taking the value 0 when no comparison can be made (because of missing values, or when $x_i = x_j = 0$ for an "asymmetric" binary variable where only the value "1" carries information). The Gower distance is produced by the daisy() function in R

(Maechler, Rousseeuw, Struyf, Hubert & Hornik, 2015), which also permits component-wise weights; we use this function in our work to compare Gower's distance with the results of the tree distance algorithm.

As part of analyzing the data, it is valuable to be able to visualize it. Data visualization is a powerful way to convey knowledge and enables decision makers to see analytic results visually. One of the most important benefits is that it makes it possible to identify and examine large amounts of data (Iliinsky, 2012). It also allows access to challenging data sets and provides useful information in an efficient way.

D. T-DISTRIBUTED STOCHASTIC EMBEDDING ALGORITHM

In this section, we describe the t-SNE algorithm for visualization. This section follows the development of van der Maaten and Hinton (2008). t-SNE stands for t-distributed stochastic neighbor embedding. The t-SNE algorithm produces a visualization of high-dimensional data by assigning individual data points into a two or three-dimensional map. The t-SNE algorithm is especially effective for high-dimensional data that consists of a large number of classes. Maaten and Hinton also explain that this algorithm is efficient not only to capture the high dimensions' local structure, but also to find a global structure having clusters with various scales. Also the algorithm produces high quality visualizations by minimizing the tendency of points to gather at the center of the map.

According to van der Maaten and Hinton (2008), the original Stochastic Neighbor Embedding (SNE) algorithm calculates Euclidean distances in high dimensions and generates conditional probabilities which reflect similarities. They set the original high-dimensional data's conditional probability as $p_{j|i}$, which is the similarity of datapoint x_j to datapoint x_i. The conditional probability for the high-dimensional data $p_{j|i}$ is defined by

$$p_{j|i} = \frac{\exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\left\|x_i - x_k\right\|^2 / 2\sigma_i^2)},$$

where σ_i is the variance of a Gaussian distribution centered on x_i. Since the density of the data varies, there is no unique optimal σ_i for all datapoints. If a part is crowded with data points, σ_i 's value is smaller than a part the data points are distant. So $p_{j|i}$ will be high for neighboring points and will be very tiny for far distant points.

They also set the low-dimensional data's conditional probability as $q_{j|i}$ for the low-dimensional analogues y_j and y_i of the high-dimensional data points x_i and x_j. The authors set the Gaussian variance σ_i to $\frac{1}{\sqrt{2}}$ for $q_{j|i}$, so the conditional probability for low-dimensional data $q_{j|i}$ is denoted by

$$q_{j|i} = \frac{\exp(-\left\|y_i - y_j\right\|^2)}{\sum_{k \neq i} \exp(-\left\|y_i - y_k\right\|^2)}$$

If the points produced for the low-dimensional map accurately represent the proximity between data points in high dimensions, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal. So the SNE algorithm is designed to find a representation of low-dimensional data points that minimizes the discrepancy between conditional probabilities.

The SNE algorithm establishes a cost function based on the sum of Kullback-Leiber divergences. The cost function C is defined by

$$C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

where P_i denotes the conditional probability distribution over all data points from x_i in high-dimensional space, and Q_i denotes the conditional probability

distribution over all data points from y_i in low-dimensional space. Since the Kullback-Leibler divergence is asymmetric, it does not measure the errors in low dimensions equally. To reduce the cost, using neighboring points is reasonable for displaying far distant points.

A gradient descent method is used for the minimization of the cost function C. The gradient has a very simple form given by

$$\frac{\delta C}{\delta y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j).$$

Although the SNE algorithm constructs reasonably good visualizations, the cost function is difficult to optimize. It also suffers from the "crowding problem." In van der Maaten and Hinton's study, the crowding problem means that the two-dimensional map is not large enough to express the distance between two points in high dimensions, so most points that are at a "moderate distance from data point i" are placed much closer than the actual distances in the high-dimensional map. So, most points that are at a "moderate distance from datapoint i" should be placed much farther apart to more accurately reflect distances in the original space. The t-SNE algorithm alleviates both these problems.

The cost function in the t-SNE algorithm differs in two ways from the cost function in the SNE algorithm (van der Maaten & Hinton, 2008). First, it uses "a symmetrized version of the SNE cost function with simpler gradients" introduced by Cook, Sutskever, Mnih, and Hinton (2007). In particular, the conditional probabilities for the high-dimensional space are replaced by

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$$

with $p_{ii}=0$ and with the analogous replacement for the conditional probabilities in the low-dimensional space.

Second, the cost function in the t-SNE algorithm uses a Student-t distribution with one degree of freedom (that is, a Cauchy distribution), while the cost function in the SNE algorithm uses a Gaussian distribution to compute the proximity between points in low dimensions.

For optimizing the t-SNE cost function, van der Maaten and Hinton (2008) suggested two more tricks. The first one is “early compression”, which means that it makes the points in the map closely gather during optimization. When two groups of mapped points are in close proximity, one cluster can move through another easily. This makes the exploration of space for global organization of the data much easier. An additional L2-penalty is added to perform “early compression” to the cost function. It is “proportional to the sum of squared distances of the map points from the origin”. The second trick is “early exaggeration,” which is to multiply all of the p_{ij}‘s by, e.g., 4 at the initial stages of the optimization. This means that almost all of the q_{ij}s, the sum of which is 1, are too small to model their corresponding p_{ij}‘s. So, the original clusters in the data produce “tight widely-separated clusters” and the resulting empty space makes clusters move around easily in order to find a good global organization.

The t-SNE algorithm attempts to preserve the data’s topology (Olah, 2014). According to the author, the algorithm defines neighboring points, “trying to make all points have the same number of neighbors.”

The t-SNE algorithm often does a good job at revealing clusters in data, but tends to get stuck in local minima (Olah, 2014). The author gives the example depicted in Figure 2 of clusters from the MNIST data set (see Section F for a description of this data). Without the color, there appear to be three clusters in Figure 2. But, points in the red cluster are separated by the blue cluster because the t-SNE algorithm converges to local minima.

Figure 2. The t-SNE local min problem on MNIST data

Source: GitHub colah/Visualizing-Deep-Learning. (2014). Retrieved from http://colah.github.io/posts/2014-10-Visualizing-MNIST/

Van der Maaten and Hinton (2008) demonstrate three potential weaknesses of their approach, even though the t-SNE algorithm outperforms other techniques for data visualization. First, the t-SNE algorithm's implementation of dimensionality reduction is obscure. This means that when the dimensionality reduction is not conducted to two or three, but to more than three dimensions, it is not known how t-SNE will perform. This problem arises because the heavy tail of the Student-t distribution comprises a large section of the probability mass in high dimensions. Second, t-SNE is sensitive to the data's inherent dimensionality due to the algorithm's local nature. The t-SNE algorithm conducts the data's dimensionality reduction on the basis of the data's local properties using a local linearity assumption on the manifold which may be violated in data sets with a high innate dimensionality. Third, the t-SNE algorithm is not assured to identify a global optimum. The non-convexity of the cost function is the main weakness of the t-SNE algorithm. The selection of several parameters is needed for optimizing and the solution depends on which parameters are selected for optimizing and initial starting conditions. According to the authors, the quality of the visualizations do not change much even with local optima.

The t-SNE algorithm is still one of the popular techniques for visualization even though it has weaknesses. We use the t-SNE algorithm for exploring dimensionality reduction.

E. DIMENSIONALITY REDUCTION

The aim of dimensionality reduction is to maintain as much of the structure in high dimensions as much as possible in the two- or three-dimensional map (van der Maaten & Hinton, 2008). That is, dimensionality reduction represents the process of remodeling high-dimensional data into low-dimensional data while assuring that the process preserves corresponding information (Ray, 2015).

Dimensionality reduction techniques reconstruct a dataset X with the original high dimension D to a dataset Y with low dimension d, preserving the structure of the dataset in high dimensions as far as possible. There are some benefits for dimensionality reduction (van der Maaten, Postma & van den Herik, 2008). First, it helps in data compression and reduces the storage space required. Second, it reduces the time required for performing the same computations. Fewer dimensions lead to less computing; they also can allow usage of algorithms unfit for high-dimensional data. Third, reducing dimensions also tends to reduce multi-collinearity among variables which in turn tends to improve the performance of statistical models fit to the data.

There are many techniques to perform dimensionality reduction. We demonstrate two common techniques here.

1. Principal Component Analysis (PCA)

Principal Components Analysis (PCA) is the one of the popular techniques for dimensionality reduction; it is also called classical multidimensional scaling. The main idea of PCA is the data points in *n*-dimensional data may lie on or near a linear subspace of dimension *d*, So given *n*-dimensional data, PCA tries to produce a subspace of *d*-dimensional data (Ghodsi, 2006). The goals of PCA are to elicit the most meaningful clue from the data, compress the data while

preserving the meaningful information, and evaluate the structure of the data set (Abdi & Williams, 2010). The PCA replaces the original variables with the principal components (linear functions of the original variables) to accomplish these goals. The first principal component is the one with the biggest variance. The second principal component has the greatest variance among those orthogonal to the first principal component. The remaining *n* components are computed likewise. Only the first *d* principal components are retained where *d* may be 2 or 3 for visualization or *d* may be chosen to be large enough to explain most (e.g. 90%) of the variability of the original variables.

PCA has a few advantages and disadvantages (Karamizadeh, Abdullah, Manaf, Zamani, & Hooman, 2013). According to the authors, the advantages of PCA are: its insensitivity to noise, reduced requirements for computer memory, and increased processing speed. The authors explain that PCA also has disadvantages. It is challenging to estimate the covariance matrix of the data, from which the principal components are derived, and PCA does not always admit of easy interpretation because each individual principal component is a linear combination of the all variables.

2. Multidimensional Scaling (MDS)

Multidimensional scaling (MDS) is one of the popular techniques for multivariate data analysis that aims to reveal the structure of a data set by plotting it in two or three dimensions. It is a powerful tool in data visualization and other data processing areas.

The goal of MDS is to find a spatial configuration in low dimensions such that the actual distance between two points, say d_{ij} , is close to the distance between the two points in the low-dimensional space after multidimensional scaling, $\hat{d}_{ij}$. The distances in the usual implementations are Euclidean. MDS arranges data points in a two- or three- dimensional map, and investigates how well the new distances between data points preserve the relationship between the high dimensional distances. Technically, it uses an algorithm that evaluates

several new arrangements and optimizes to maximize the goodness-of-fit (Sahasrabudhe, Machiraju, & Zhu, 2001).

Equivalently, according to van der Maaten, Postma and van den Herik (2008), the stress measures the quality of the mapping by measuring the error between the low-dimensional data's pairwise distances and the high-dimensional data's pairwise distances. When the distances are Euclidian, the raw stress function for MDS is given by

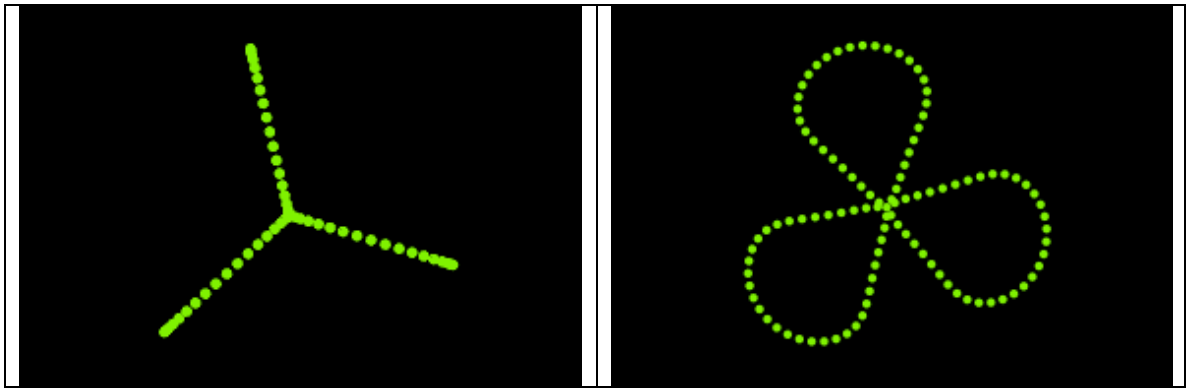
$$\text{cost} = \sum_{ij} \left(\left\| x_i - x_j \right\| - \left\| y_i - y_j \right\| \right)^2, \text{ where}$$

$||x_i - x_j||$ is the Euclidean distance between the high-dimensional data points and $||y_i - y_j||$ is the Euclidean distance between the low-dimensional data points.

MDS is a broad term that includes several types of mappings. The types include metric and non-metric MDS and CMDS (Young, 1985).

One example of non-metric MDS is Sammon mapping. It attempts to "minimize the differences between corresponding inter-point distances in the two spaces", which are the original high-dimensional one and the low dimensional one, and tries to preserve structure in high dimensions (Henderson, 1997). The author gives projection pictures (Figure 3) to compare PCA with Sammon mapping. The data set has "three mutually perpendicular circles" in six-dimensional space. The left side picture, produced by PCA, shows that the technique does not preserve the circles in the two-dimensional mapping. In contrast the right side picture, produced by Sammon mapping, shows some of the topology of the original data set.

Figure 3. PCA and Sammon projection of six-dimensions



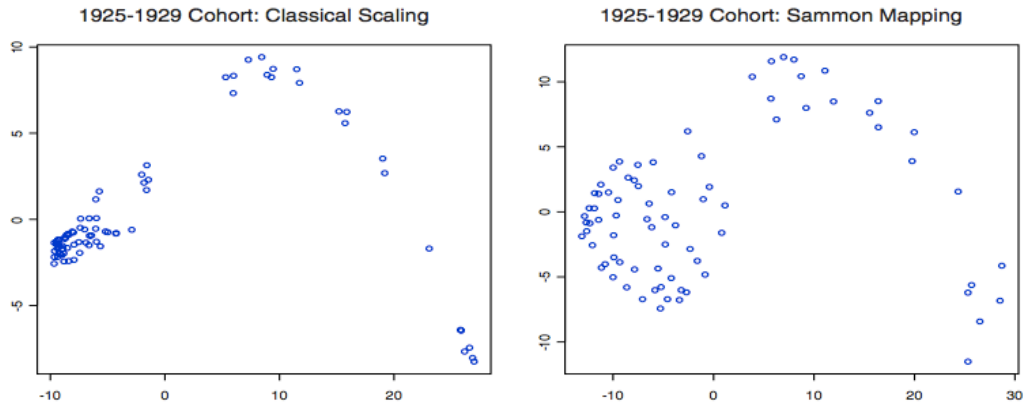
Source: Sammon mapping. (1997). http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0910/henderson.pdf

According to the author, the stress for Sammon mapping, defined as

$$E = \frac{1}{\sum_{i<j} d_{ij}^{*}} \sum_{i<j}^{n} \frac{(d_{ij}^{*} - d_{ij})^2}{d_{ij}^{*}}, \text{ where}$$

d_{ij} is the pairwise distance between data points in low-dimensional space and d_{ij}^{*} is the pairwise distance between data points in high-dimensional space. Sammon mapping accepts d_{ij}^{*} as Euclidean distance and keeps small d_{ij}^{*} since it gives a higher degree of importance to small d_{ij}^{*} (Jung, 2013). Figure 4 shows the results of the 1925-1929 cohorts of the bank employee data (analyzed in Izenman, 2008). It displays the CMDS in the left panel and the Sammon mapping in the right. The Sammon mapping preserves small d_{ij}^{*}s better than CMDS, while compressing relatively larger d_{ij}^{*}s.

Figure 4. Classical multidimensional scaling and Sammon mapping



Source: Multidimensional scaling, (2013). Retrieved from http://www.stat.pitt.edu/sungkyu/course/2221Fall13/lec8_mds_combined.pdf

Sammon's non-linear mapping is implemented through the sammon() function in R's MASS library (Venables & Ripley, 2002). We use this function for visualization as one of the MDS techniques.

Another non-metric MDS is Kruskal's non-metric MDS, which is implemented in the isoMDS() function in R. It uses the stress function, defined as

$$stress(1) = \sqrt{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2 / \sum_{i<j} d_{ij}^2}, \text{ where}$$

d_{ij} is the actual distance and $\hat{d}_{ij}$ is the distance in lower-dimensional space (Izenman, 2008).

We also use CMDS for visualization to compare to the t-SNE algorithm. CMDS arranges the data points in a low-dimensional map to reduce the discrepancy between the pairwise distances in high dimensions and the pairwise distances in low dimensions. CDMS finds the centered configuration $x_1, \ldots x_n \in \mathrm{R}^q$ for some $q \geq n-1$ so that their pairwise distances are the same as the original distances; then dimensionality reduction from $X = X_q$ to X_p $(p < q)$ proceeds as in principal component analysis (Jung, 2013).

One problem of MDS is that its complexity increases quickly with the number of dimensions. This increase in the number of parameters means that the resulting model can be as complex as the data itself. Even though MDS has difficulties, it is still one of the popular dimensionality reduction techniques. It performs particularly well on relatively small data sets (Young, 1985).

F. DATA SETS

In this work we produce Gower and tree distance measures of inter-point dissimilarity in high dimensions. Then we apply the Barns-Hut implementation of the t-SNE algorithm (Krijthe, 2015), CMDS (R Core Team, 2015), and non-metric MDS (Venables & Ripley, 2002) to those distances to determine combinations that produce consistently good visualizations. In this section, we describe the characteristics of the data sets used in our work. Each of the data sets has a known class variable which is not incorporated into the inter-point distances. One measure of whether the visualization of the data is adequate is whether observations from different classes tend to fall in different clusters in the low-dimensional map.

1. Splice

The Splice data is taken from the UC Irvine Machine Learning Repository (Lichman, 2013). This database's original name is "primate splice-junction gene sequences" data set. All samples are taken from Genbank 64.1. The Splice data has been widely used for machine learning techniques. The number of instances is 3190 and the number of attributes is 62, which consist of the instance name, 60 sequential DNA nucleotide positions and the class. Attribute number 1 (V1) is one of {N, EI, IE}, indicating the class. IE denotes a "from intron, which are the parts of the DNA sequence that are spliced out, to exon, which are the parts of the DNA sequence retained after splicing" boundary; EI denotes a "from exon to intron" boundary, and N means "neither." Attribute number 2 (V2) is the instance name and is removed. Attribute numbers 3 to 62 are the sequence and each of these attributes is usually filled by one of {A, G, T, C}. Other characters {D, N, S,

R} imply imprecise knowledge among the characters {A, G, T, C}, so we do not use the observations which include the four characters {D, N, S, R} for our test. After excluding these observations and withholding the class, this data set has 3,175 instances and 60 attributes.

2. MNIST

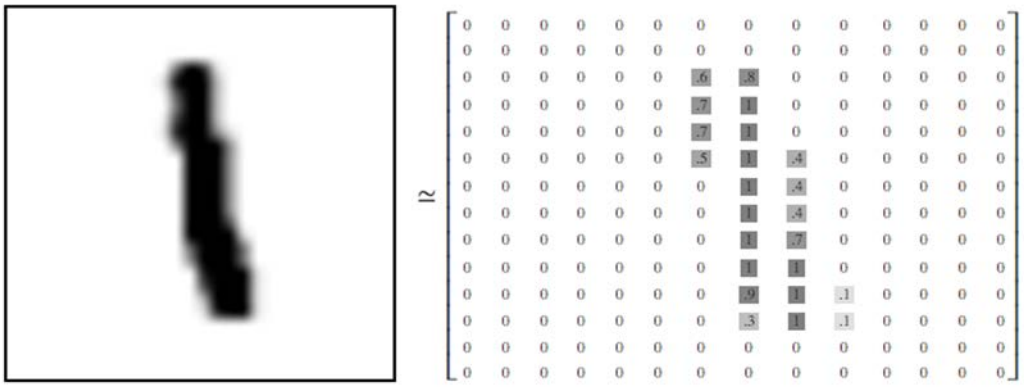
We took the MNIST data from Yann LeCun's website (LeCun, 2016). It is data on a large set of handwritten digits data for a digit recognition system. The training set has 60,000 digits each representing a number from 0 to 9 and the test set has another 10,000 digits. Each monochrome image has 28 by 28 pixels, which is 784 pixels total, and is centered within a box (Olah, 2014). Figure 5 contains examples of the MNIST data sets.

Figure 5. Examples of MNIST data set

Source: Christopher Olah. (2014). "Visualiing MNIST: An exploration of dimensinality reduction," October 9. Retrieved from http://colah.github.io/posts/2014-10-Visualizing-MNIST/

According to the author, MNIST is a simple computer vision dataset. As mentioned above, MNIST data consists of 28x28 pixel images of handwritten digits. So the image can be regarded as "an array of numbers describing how dark each pixel is". For instance, we can think of number 1 as in Figure 6. Figure 6 shows how the pixels correspond to the numbers' appearance.

Figure 6. Examples of MNIST data set



Source: Christopher Olah. (2014). “Visualiing MNIST: An exploration of dimensinality reduction,” October 9. Retrieved from http://colah.github.io/posts/2014-10-Visualizing-MNIST/

As we can see in Figure 6, there is a 28 by 28 array for each image in MNIST data; this can be unfolded into a 784-dimensional vector for each observation. The vector’s value indicates “how dark” the pixel is and the value is between zero and one (Olah, 2014).

MNIST is a favorable data set for learning pattern recognition and other techniques, because we do not need to spend much time and effort to process and format the data (LeCun, Bottou, Bengio, & Haffner, 1998). Practically, the MNIST data is used vigorously for machine learning and neural networks today.

We use the MNIST data set for our experiment, because it is well processed and formatted, as it is mentioned above, and it is a relatively large data set which has 784 dimensions. In practice we often use a sample of 1,000 records or so, rather than using the entire set of 60,000 records.

3. Covertype

The Covertype data set is taken from the UC Irvine Machine Learning Repository (Lichman, 2013). This database’s original name is “forest cover type dataset” and initially compiled by Jock A. Blackard. It is for predicting forest cover type only from cartographic variables and a mixed-type data set. The data set

has 54 variables, of which ten are quantitative measures and 44 are binary variables representing soil conditions and wilderness areas (Meyer, 2001). The response variable is the forest cover type, which are seven specific forest cover types; spruce/fir, lodgepole pine, ponderosa pine, cottonwood/willow, aspen, douglas-fir, and krummholz. The actual forest cover type and the other variables are from US Forest Service and US Geological Survey. The total number of observations is 581,102 and the training set includes 11,340. We sample 1,000 rows and use this mixed data as our third data set.

G. SUMMARY

In this chapter, we reviewed the characteristic of big data sets today and clustering, which is an important tool in the analysis of big data. We reviewed the tree distance algorithm that we use to measure inter-point distances in our data sets. The tree distance algorithm has benefits for mixed data type, noise, outliers, and different scales of data. Then we reviewed the t-SNE algorithm for our visualization. The t-SNE algorithm is a popular visualization technique, especially for high-dimensional data. We note that categorical variables with c classes are represented by c or c–1 binary variables, thus even data sets containing a moderate number of categorical variables can be thought of as high-dimensional data. And we reviewed dimensionality reduction and some common techniques. At the end of the chapter, we described the data sets we used in our research: the Splice, the MNIST, and the Covertype data sets.

III. METHODOLOGY

In this chapter, we demonstrate: the tree distance algorithm for computing inter-point distances, the Barnes-Hut implementation of the t-SNE algorithm and CMDS for visualization and dimensionality reduction. Then, we describe the experimental setup for our dimensionality reduction experiment. The computation time of the Barnes-Hut t-SNE algorithm is much less expensive than that of the original t-SNE algorithm and also outperforms it on mapping data from high dimensions to low dimensions. We demonstrate how we conduct the new dimensionality reduction technique in this chapter. The chapter is organized as follows: Section A describes the treeClust package in R. Section B demonstrates the Rtsne package in the R for t-SNE visualization. Section C describes CMDS. Section D introduces how we explore the new technique for dimensionality reduction.

A. TREECLUST ALGORITHM FOR CLUSTERING

In this section, we describe the treeClust package in R we use for computing inter-point distances. We use the tree distance algorithm implemented using treeClust for clustering since Euclidean distance usually needs to be extended when some of the attributes are categorical (Buttrey & Whitaker, 2015b). The package has also an ability to generate a new numeric data set, which is called “newdata,” which has the property that the inter-point distances among observations in “newdata” mirror the inter-point distances computed with the treeClust mechanism. This feature of treeClust allows us to handle larger data sets, since the “newdata” set will generally have fewer entries than the matrix of all pairwise inter-point distances produced by, for example, the Gower technique.

Some features of treeClust deserve mention here. First, there is a choice of tree-based dissimilarity measure, indicated by an integer from 1 to 4 and we apply 4. Buttrey and Whitaker (2015a) compared the clustering method’s

performance with Cramer's *V*, and dissimilarity measure 4 frequently showed the Cramer's *V* value higher than that of the other measures. Second, a control argument allows us to modify some of the parameters to the algorithm and to determine which results should be returned. For example, the user can request the "newdata" object, which is computed not from pairwise distances among observations, but from the set of pairwise distances among leaves (Buttrey & Whitaker, 2015b).

We apply both Gower and tree distance approaches to include both categorical and numeric values; then we use the CMDS algorithm and the t-SNE algorithm to the pairwise distances (Gower) or the "dists" (treeClust) for exploring visualization and dimensionality reduction.

B. BARNES-HUT T-SNE ALGORITHM FOR VISUALIZATION

In this section, we describe the Barnes-Hut implementation of the t-SNE algorithm. Krijthe (2015) provides this implementation in the Rtsne package in R. According to van der Maaten (2014), the computational complexity of the SNE class of algorithms for "the number of input objects *N*" increases exponentially and it is the main limitation of the t-SNE algorithm. Practically, the application of the t-SNE algorithm is limited to relatively small data sets, with only a few thousand points. The author explored the Barnes-Hut approximation for the SNE class of algorithms that "require only $O(N \log N)$ computation and $O(N)$ memory." Application of Barnes-Hut to the t-SNE algorithm shows that the algorithm is considerably accelerated compared to the standard t-SNE algorithm, and it visualizes the large data sets successfully as well.

In practice, we examined the tsne package in R, Donaldson (2012), for the Splice and MNIST data sets at first. But we found that the tsne package requires much more computation time than the Rtsne package, which uses the Barnes-Hut t-SNE algorithm. For example, running the tsne() function (from the tsne package) on a sample of 1,000 observations from the MNIST data, required 390 seconds. On the other hand, the Rtsne() function on the same data required only

32 seconds, less than a tenth of the time. Therefore we used the Barnes-Hut t-SNE algorithm, in the Rtsne package in R, instead of the original t-SNE algorithm, from the tsne package. The Barns-Hut t-SNE algorithm is also robust for distinguishing classes of large data set in terms of visualization.

Figure 7 and Figure 8 shows the 2D plots for the sample of 1,000 observations from the MNIST data. In each plot the points are labeled and colored by the correct classification (that is, the actual digit written). It appears that the plot for Rtsne (Figure 8) seems more useful in distinguishing the classes than the plot for tsne (Figure 7).

Figure 7. t-SNE 2D plot of MNIST data

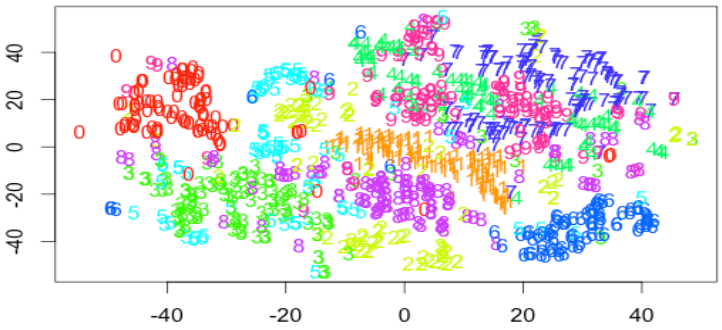
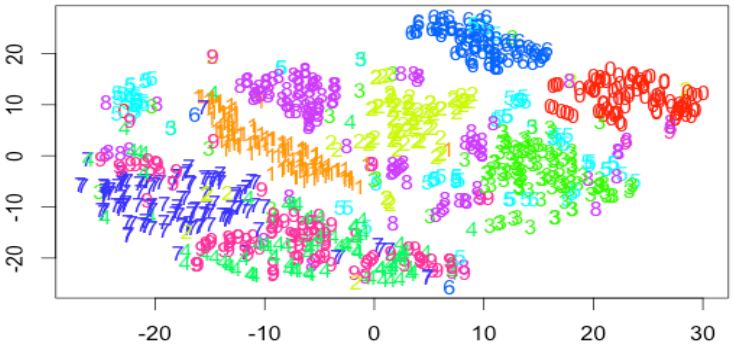


Figure 8. Rtsne 2D plot of MNIST data



We also sampled 500 observations from the Splice data and applied the Rtsne function using tree distance because all Splice variables are categorical. Figure 9 is the 2D plot using Rtsne for Splice data with each observation colored by its true class. The points in Figure 9 overlap a lot, so we cannot determine easily whether the t-SNE can separate the true classes or not. So, we plotted a three-dimensional t-SNE mapping using Rtsne in Figure 10.

Figure 9. Rtsne 2D plot of Splice data

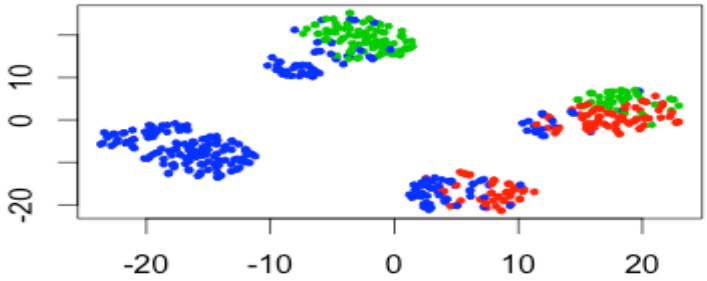
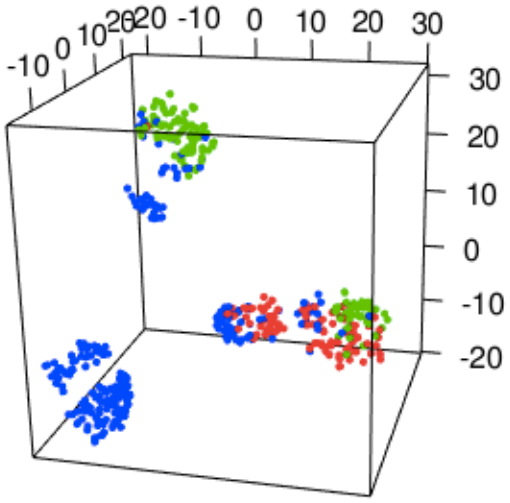


Figure 10. Rtsne 3D plot of Splice data



The three-dimensional version outperforms the two dimensional one, especially in terms of the extent of overlapping. Because the t-SNE algorithm tries to put a lot of space between clusters, the points are mapping crowded inside the clusters. We explore two dimensions and three dimensions together to see how the dimensionality reduction performs over the overlapping part as well.

C. CLASSICAL MULTIDIMENSIONAL SCALING (CMDS)

We compare CMDS with the results of the Barnes-Hut t-SNE algorithm. CMDS is the one of the traditional dimensionality reduction techniques and it is a linear technique that tries to keep the representation of dissimilarity between two points in low dimensions far apart (van der Maaten & Hinton, 2008).

We also sampled 500 observations from the Splice data, and use CMDS, which is implemented in the cmdscale() function in R. Figure 11 is the picture of the result when we applied the CMDS to the Splice sample using tree distance. The result looks quite good even though the points overlap a little. We also plotted the three-dimensional picture (Figure 12).

Figure 11. CMDS 2D plot of Splice data

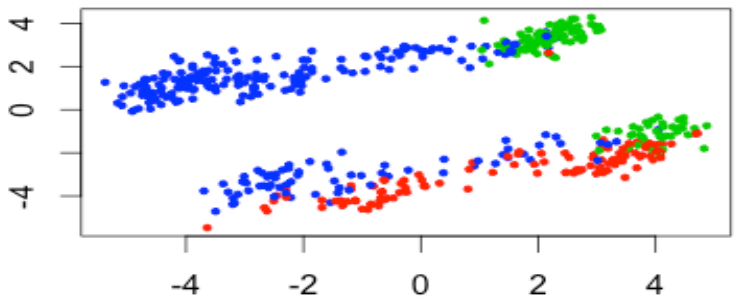
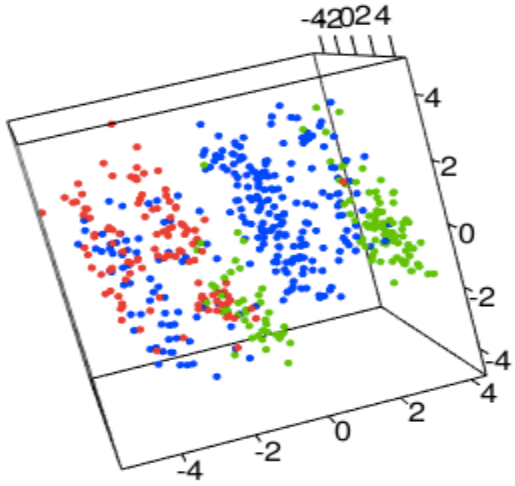


Figure 12. CDMS 3D plot of Splice data



We can see the result of CMDS more clearly in three-dimensional plot. We also use the Gower distance, which is implemented by the daisy() function in R, for clustering and compare the performances both of the two distances, Gower and tree distance for clustering and of the two visualization techniques, which are implemented by the functions cmdscale() and Rtsne().

D. EXPERIMENTS

In this thesis, we compare the Barnes Hut t-SNE algorithm and CMDS using Gower distance and tree distance respectively and evaluate our dimensionality reduction experiment for the t-SNE algorithm.

Generally, CMDS performs well for visualization and dimensionality reduction. But if the data set has a lot of variables and is of mixed data type, it can produce poor pictures. The t-SNE algorithm frequently performs better, but its performance depends on the inter-point distance used. The Gower distance is widely used in clustering, while the tree distance is robust for mixed data and outliers. So we compare two visualization and clustering based on two distances and explore which one performs better.

Moreover, dimensionality reduction techniques usually try to map the data from high dimensions to two or three dimensions directly. We explore a new technique that does not appear to have been tried in the literature. We conduct what we call “longer path dimensionality reduction” using Barnes Hut t-SNE algorithm on a data set, starting with a very high, original dimensionality from (perhaps 100 or 200) to a high dimensionality (e.g., 60, 50) to a moderate number of dimensions (e.g., 30,10) to a low number of dimensions (e.g., 3, 2). We explore this technique on the Splice data, which is relatively small data set and categorical, to the MNIST data, which is relatively large data set and numerical, and to the Covertype data, which is a large data set of mixed type – although for computational reasons, and to keep pictures from being overrun with points, we use samples in these last two cases.

To recap, then, for each data set, we withheld the class variable and used it only to color or label points in the pictures. We sampled 3,000 records from MNIST and 1,000 records from Covertype. We use the daisy() function to compute the Gower distance, and implemented the classical multidimensional scaling technique, Sammon mapping, the isoMDS algorithm, and Barnes-Hut t-SNE using the R function cmdscale(), sammon(), isoMDS() and Rtsne(). Then we used the treeClust() function to compute inter-point distances and implement the same visualization techniques. We show the resulting mappings as two- or three-dimensional pictures and add color to the points based on class to identify how well the mapping preserves classes.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS

In this chapter, we describe the results of using several visualization techniques with the Gower distance and the tree distance and the results of our experiment for dimensionality reduction. We demonstrate the results of our three data sets in sections A, B, and C.

A. THE RESULTS WITH THE SPLICE DATA SET

As described above, we computed the Gower and tree distances in order to compare those two techniques. For each distance measurement we conducted CMDS (R Core Team, 2015), non-metric MDS (isoMDS) (Venables & Ripley, 2002), Sammon mapping (Sammon MDS) (Venables & Ripley, 2002) and the t-SNE algorithm on Splice data set.

There are some cases where the three-dimensional plot displays much more informatively and makes the structure of the data easier to understand than the two-dimensional plot does. But, sometimes the two-dimensional plot produces clearer visualizations. So we produce both plots for a better understanding of our experiment. Also, we conducted “long path” dimensionality reduction with MDS, but the plots look just about the same as the plot without taking long path dimensionality reduction. So only the t-SNE algorithm was used for long path dimensionality reduction.

The Splice data set has 3175 rows and 60 variables. Figure 13 shows the 2D plots for CMDS, isoMDS, Sammon MDS, and t-SNE using Gower distances. It appears that CMDS performs better in distinguishing the classes (colored dots) than t-SNE. We can see the results more clearly in the 3D plots (Figure 14).

Figure 13. Splice data 2D using daisy() function

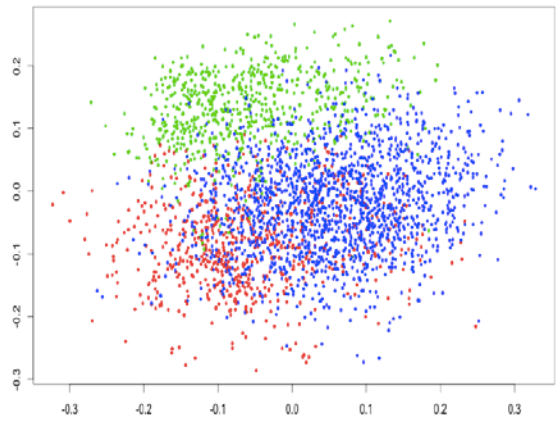
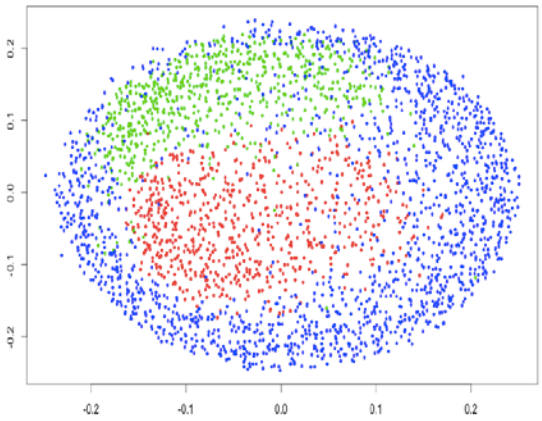
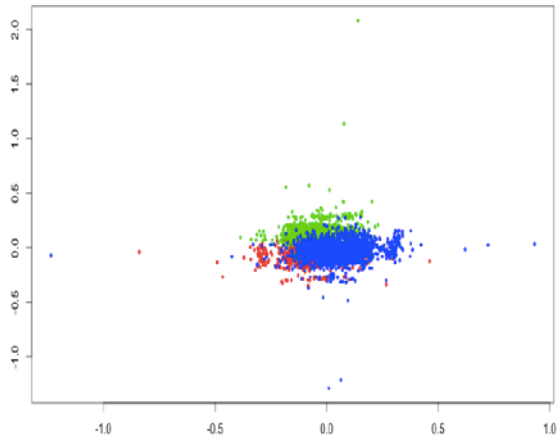
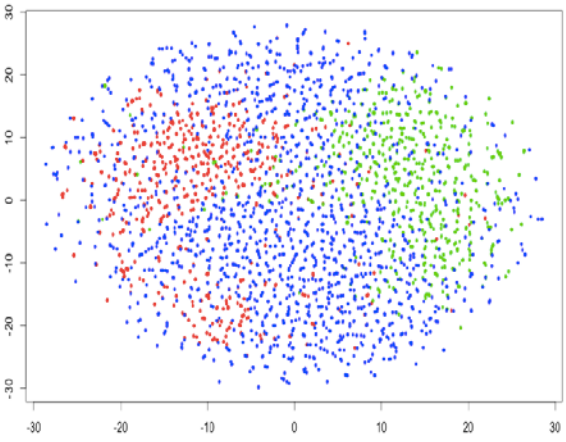
cmdscale()	isoMDS()
	
Sammon()	**Rtsne()**
	

Figure 14. Splice data 3D using daisy() function

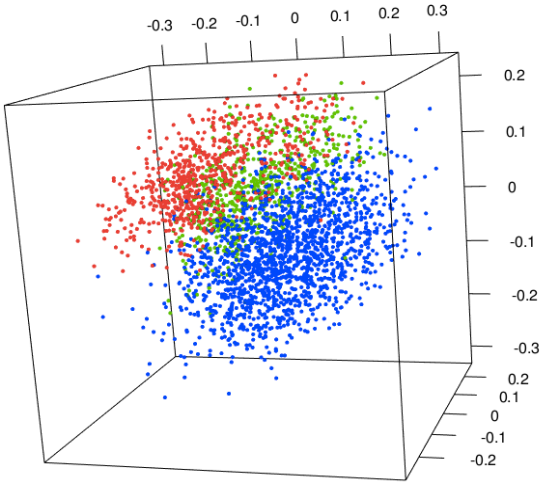
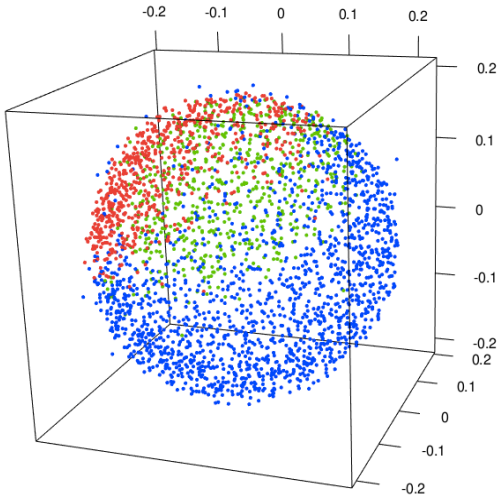
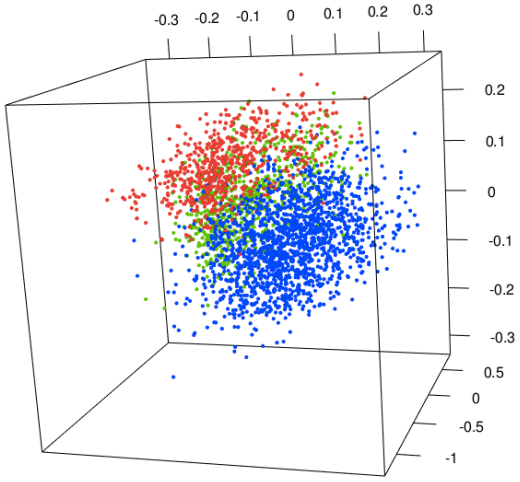
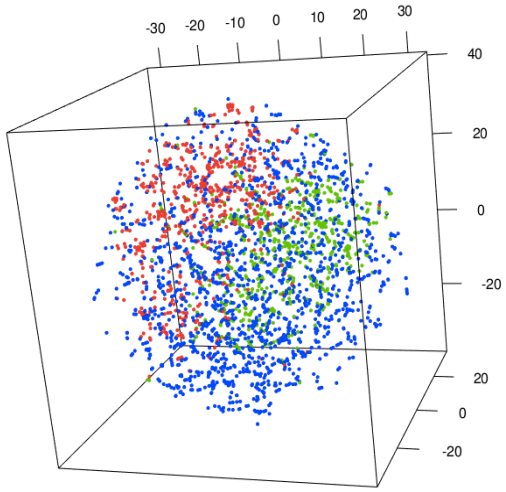
cmdscale()	isoMDS()
	
Sammon()	**Rtsne()**
	

Figure 15 shows the result of taking the long path to dimensionality reduction using the Rtsne() function with the Gower distances. We can see the 3D plots as well (Figure 16). The plots do not look as good as the CMDS plot. Some observations are overlapped and some boundaries between two classes are ambiguous.

Figure 15. Long path of t-SNE of Splice data using daisy() function

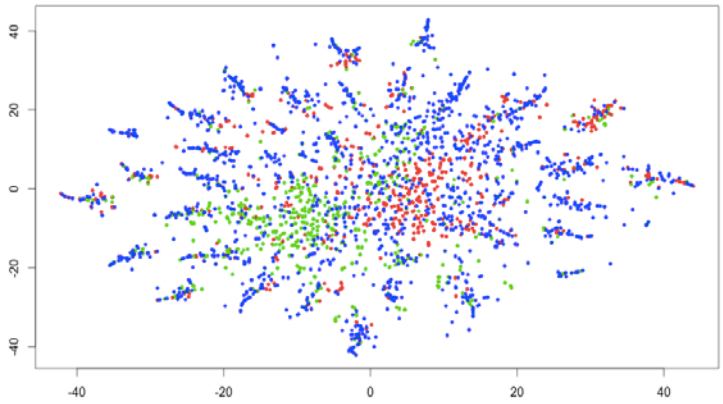
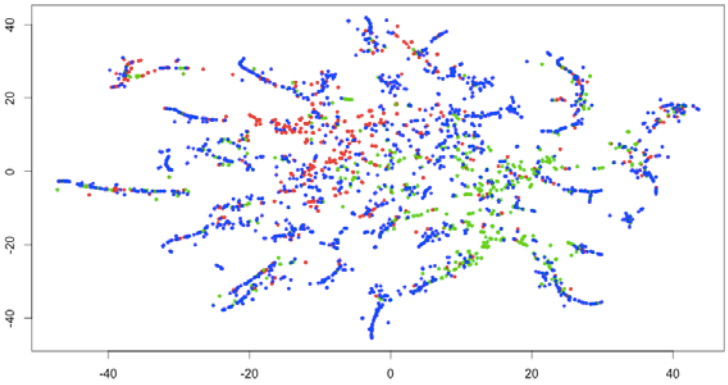
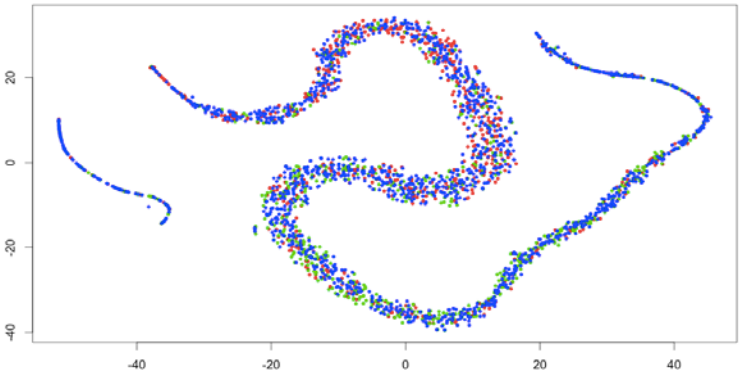
From original dimension to 60 to 2

From original dimension to 60 to 60 to 2

From original dimension to 60 to 50 to 2


Figure 16. Long path of t-SNE of Splice data using daisy() function

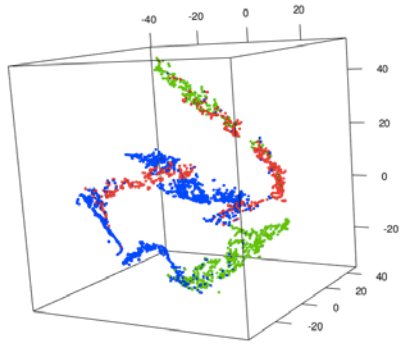
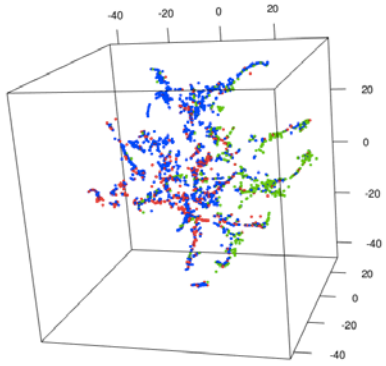
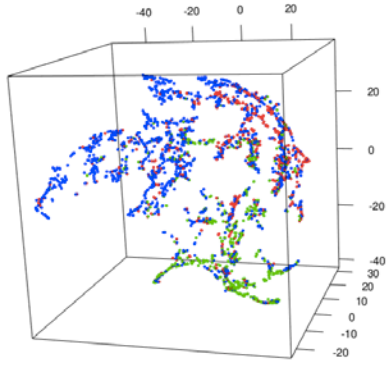
From original dimension to 60 to 3

From original dimension to 60 to 60 to 3

From original dimension to 60 to 50 to 3


Figure 17 is the 2D plots of CMDS, isoMDS, Sammon MDS, and t-SNE using the treeClust() function. They look quite different from the plots using daisy() function. CMDS plot still looks good, and is divided into several clusters that clustered more specifically. The t-SNE plot for treeClust() has a little overlap, but is much better than the one for daisy(). We can see that the treeClust() performs well for clustering in the 3D plot as well (Figure 18).

Figure 17. Splice data 2D using treeClust() function

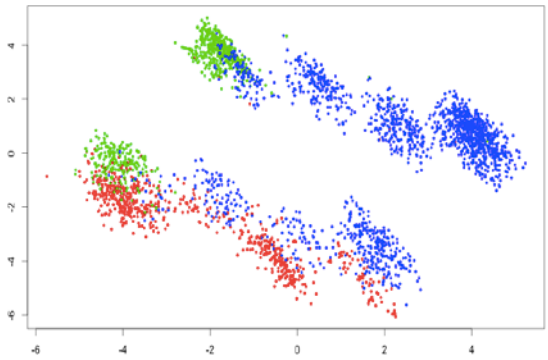
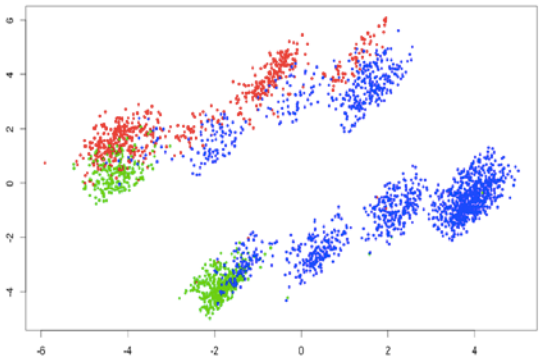
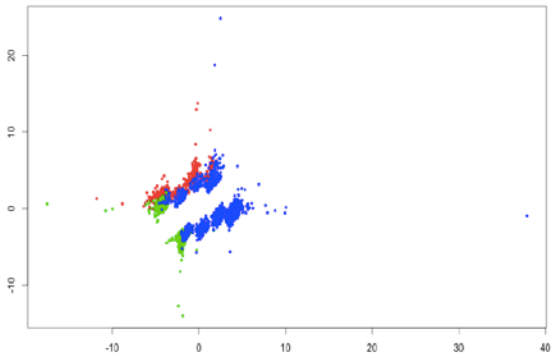
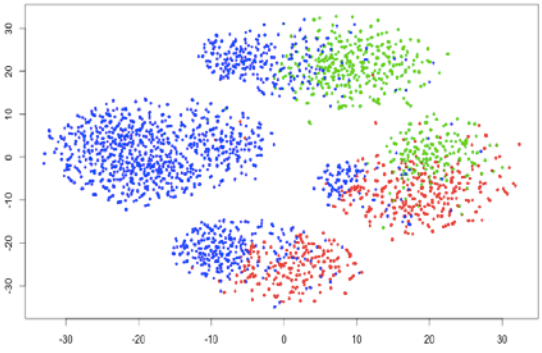
cmdscale()	isoMDS()
	
Sammon()	**Rtsne()**
	

Figure 18. Splice data 3D using treeClust() function

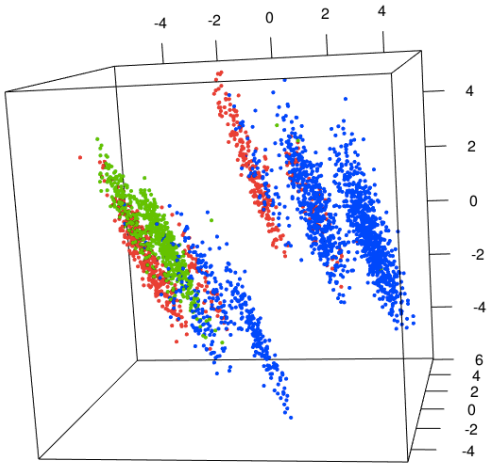
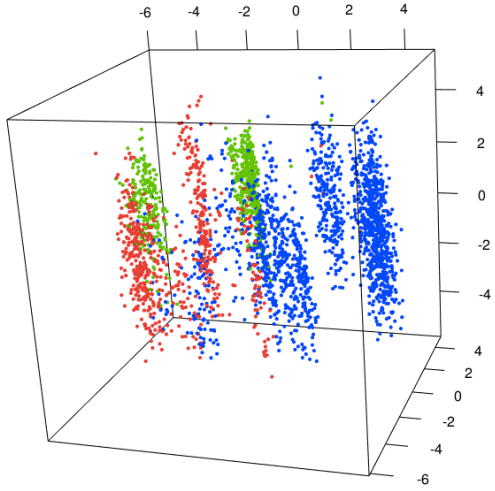
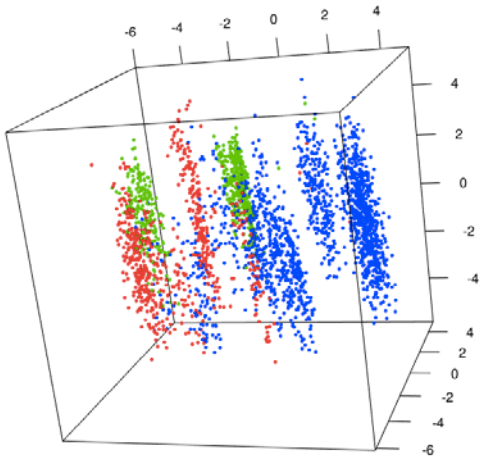
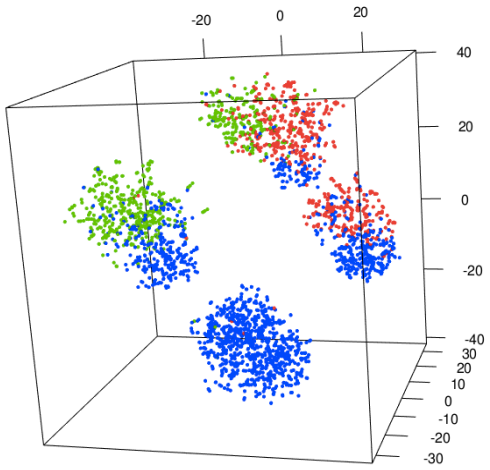
cmdscale()	isoMDS()
	
Sammon()	**Rtsne()**
	

Figure 19 shows the plots of taking long path dimensionality reduction using the treeClust() function. It looks much better than the plots using the daisy() function as well. The shape is a little twisted, but the picture separates the classes more clearly. This result is visible in the 3D plots, too (Figure 20).

Figure 19. Long path of t-SNE of Splice data using treeClust() function

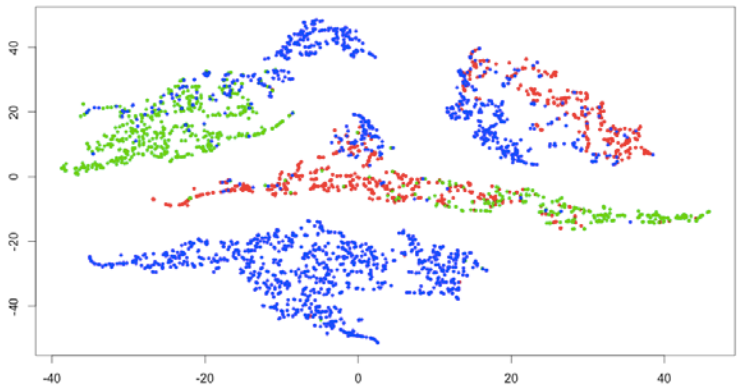
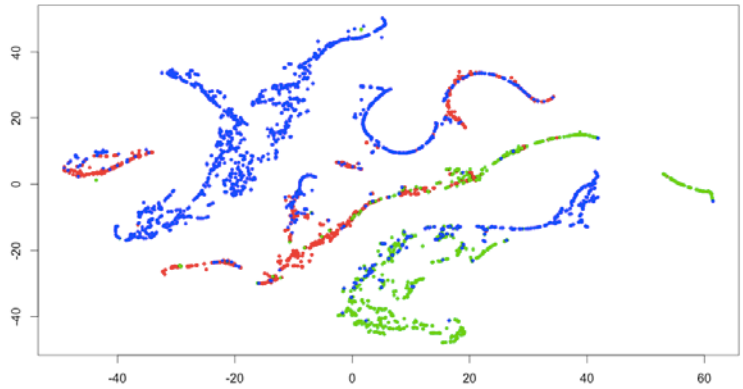
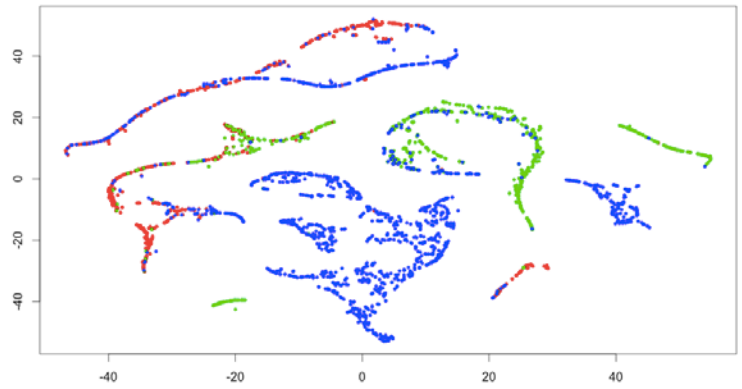
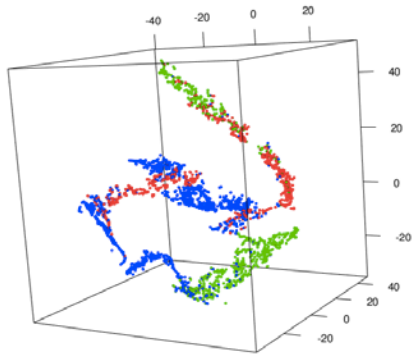
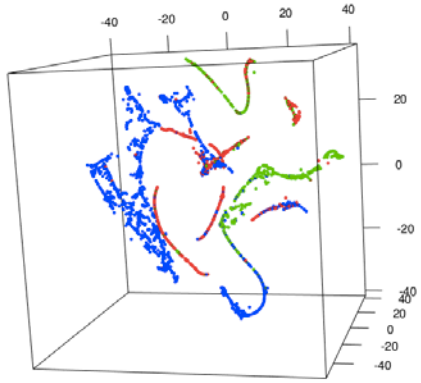
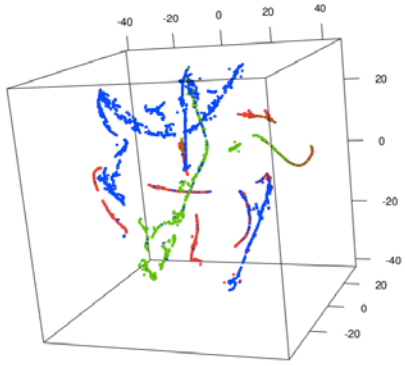
From original dimension to 60 to 2

From original dimension to 60 to 60 to 2

From original dimension to 60 to 50 to 2


Figure 20. Long path of t-SNE of Splice data using treeClust() function

From original dimension to 60 to 3

From original dimension to 60 to 60 to 3

From original dimension to 60 to 50 to 3


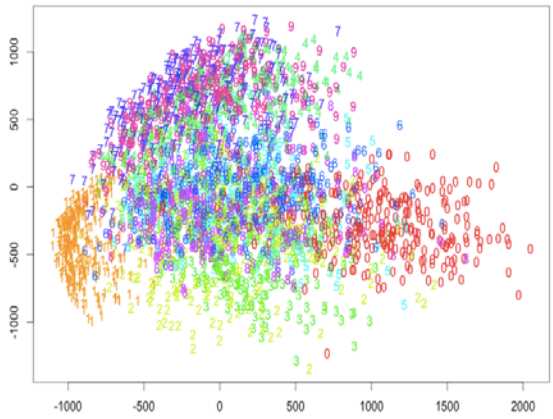
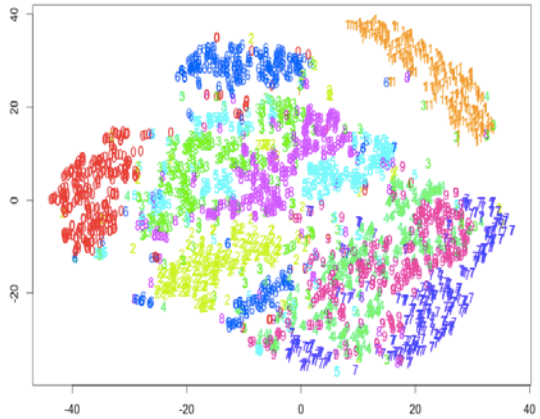
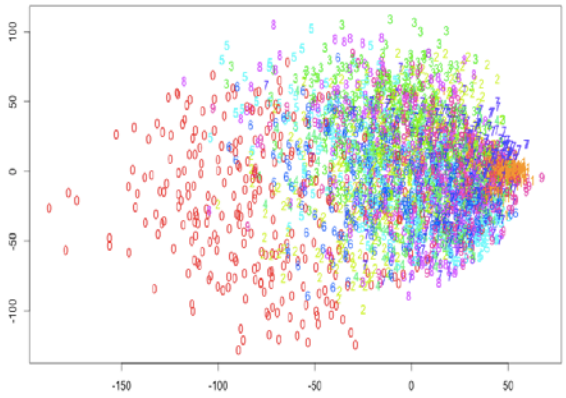
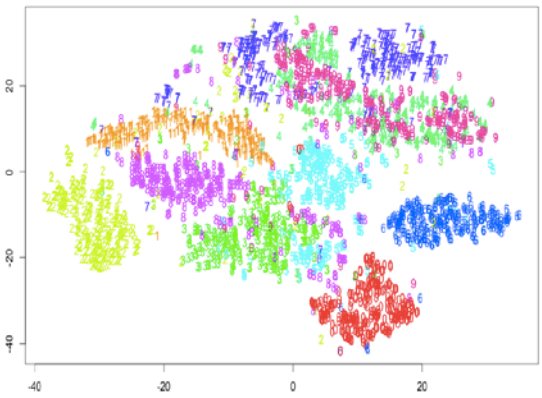
B. THE RESULTS WITH THE MNIST DATA SET

Our second data set is the MNIST data. It is quite a large data set, having 60,000 rows in the training data. So we sampled 3,000 points and explored clustering, visualization and long path dimensionality reduction. For this data set and the next we focus on the more successful CMDS and omit the results from the Sammon and isoMDS mappings.

We found that there are some computation problems with taking the long path to dimensionality reduction. An unknown computation error occurred in the Rtsne package when we tried to select a dimensionality under fifty but greater than three. Errors occurred with 40, 30, 20, and ten dimensions, so we concluded that the Rtsne algorithm does not operate properly for fewer than fifty dimensions and explored a limited dimensionality reduction. The paths we tried were from the original dimension to 60 to 2 (or 3) dimensions, from the original to 60 to 60 to 2 (3), and from the original to 60 to 50 to 2(3). There were also issues for dimensionality greater than sixty.

Figure 21 shows the plots of CMDS and t-SNE using the daisy() and treeClust() function respectively. Surprisingly, CMDS plots look agglomerated; we can barely recognize the classes unlike in the Splice data. For t-SNE, the groups look well-separated for both daisy() and treeClust(). The plot for treeClust() displays boundaries between classes more obviously than the plot for daisy() and there are some overlapped parts in the plot for daisy() – although the plot for daisy() is informative too.

Figure 21. MNIST data 2D using daisy() and treeClust() function

cmdscale() using daisy()	Rtsne() using daisy()
	
cmdscale() using treeClust()	**Rtsne() using treeClust()**
	

Figures 22 and 23 shows the plots from the long path dimensionality reduction using the daisy() function. The plot taking the longer path, e.g., from original to 60 to 50 to 2, appears to capture the clusters more obviously. We also found the interesting picture of t-SNE algorithm when we take long path dimensionality reduction. We have not figured out why, but the t-SNE algorithm tends produce twisted shapes when we take long path dimensionality reduction. Figures 24 and 25 plot the long path dimensionality reduction using the treeClust() function. The cluster boundaries in the plots using the treeClust() function look more obvious.

Figure 22. Long path of t-SNE of MNIST data using daisy() function

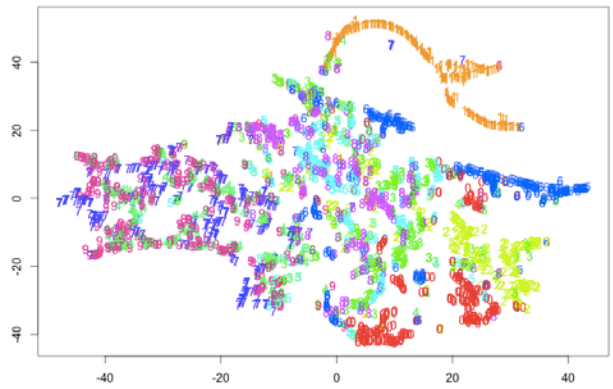
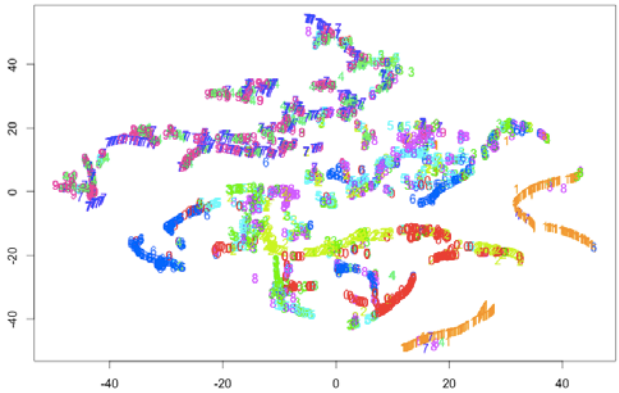
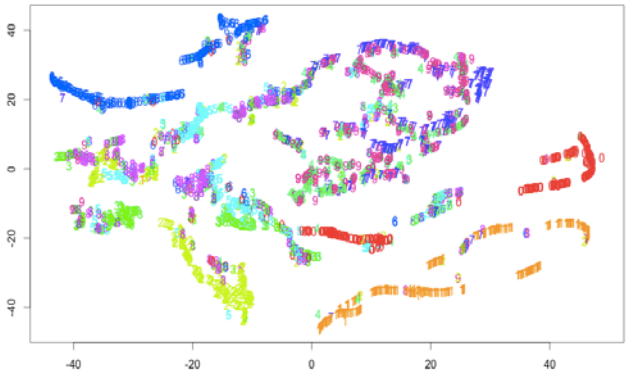
From original dimension to 60 to 2

From original dimension to 60 to 60 to 2

From original dimension to 60 to 50 to 2


Figure 23. Long path of t-SNE of MNIST data using daisy() function

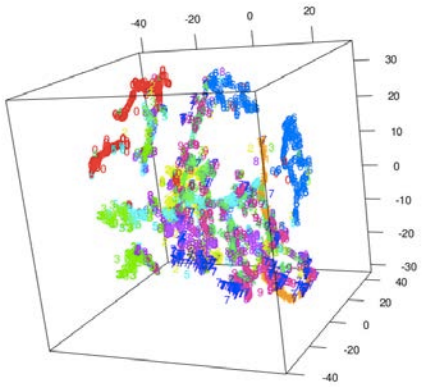
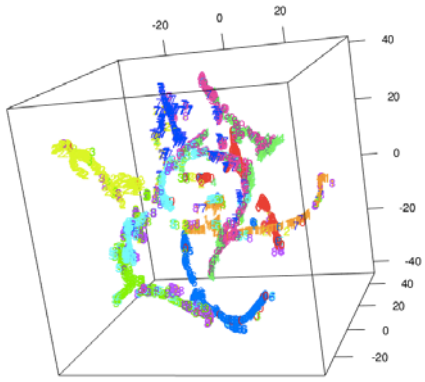
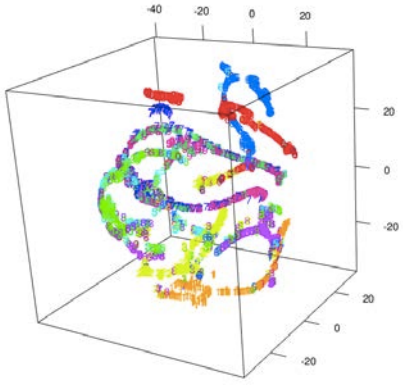
From original dimension to 60 to 3

From original dimension to 60 to 60 to 3

From original dimension to 60 to 50 to 3


Figure 24. Long path of t-SNE of MNIST data using treeClust() function

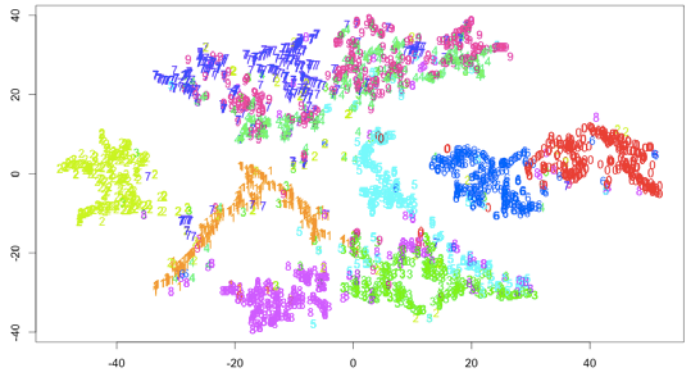
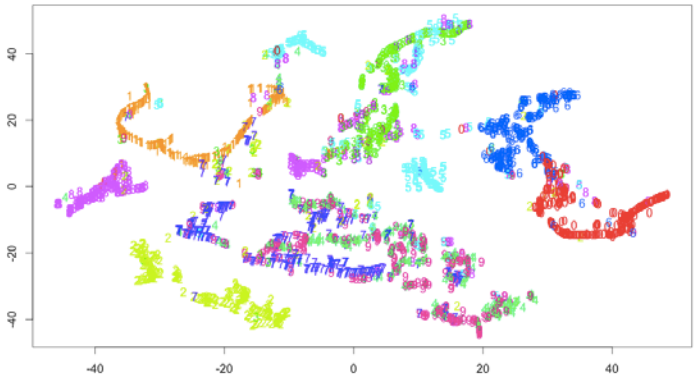
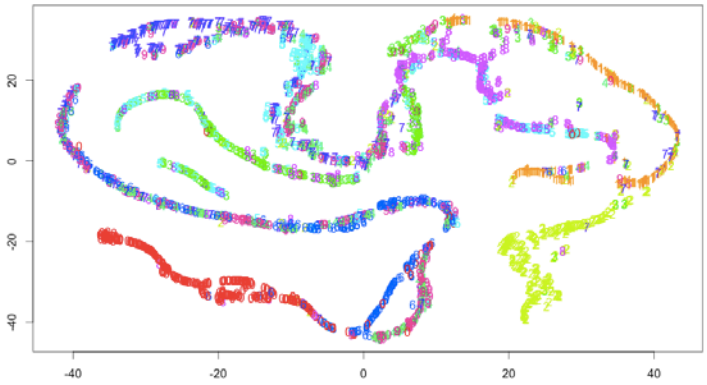
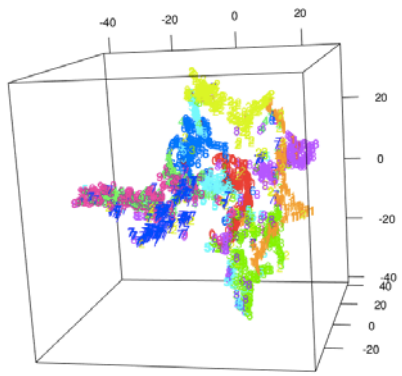
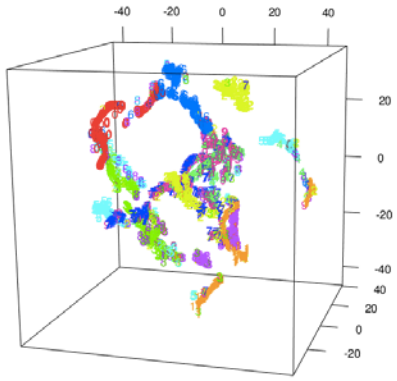
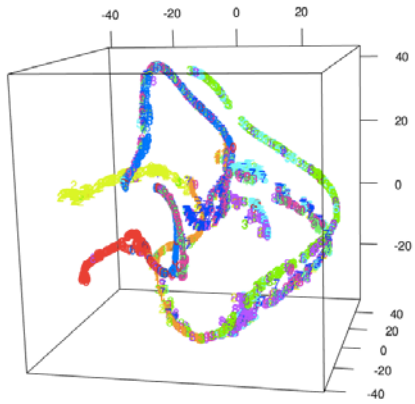
From original dimension to 60 to 2

From original dimension to 60 to 60 to 2

From original dimension to 60 to 50 to 2


Figure 25. Long path of t-SNE of MNIST data using treeClust() function

From original dimension to 60 to 3

From original dimension to 60 to 60 to 3

From original dimension to 60 to 50 to 3


C. THE RESULTS WITH THE COVERTYPE DATA SET

Our third data set is the Covertype data. It is also quite a large data set, with 11,340 rows in the training set, and mixed—both numerical and categorical—variables. So we sampled 1,000 points and explored clustering, visualization and long path dimensionality reduction to see how they work for mixed type data set.

Figure 26 gives the plots of CMDS and t-SNE using daisy() and treeClust() functions respectively. The plots show more apparent distinction between daisy() and treeClust() function. As we described, the tree distance algorithm, which is implemented as the treeClust() function, is robust to outliers, missing values, various scales, and mixed type data, while Gower distance is not. Certainly, treeClust() function outperforms daisy() function, especially for this mixed type data set. So we conclude that the combination of treeClust() for clustering and Rtsne() for visualization can produce good results in mixed-type data sets.

Figure 26. Covertype data using daisy() and treeClust() function

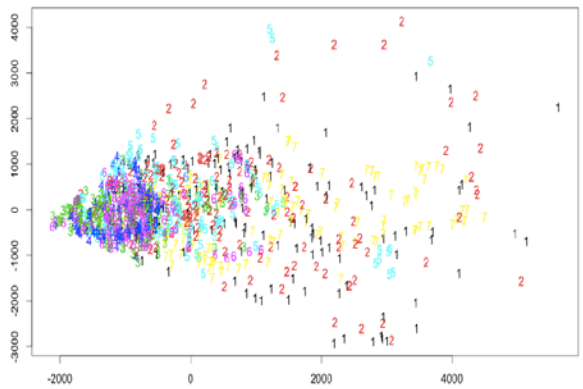
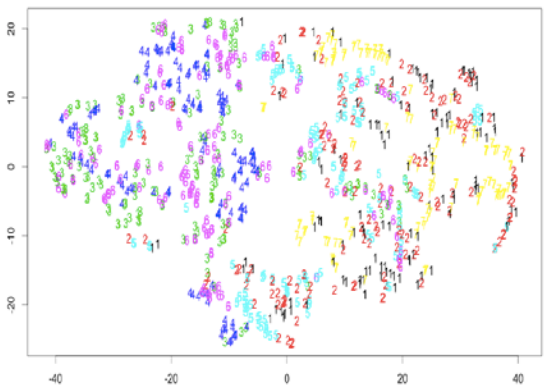
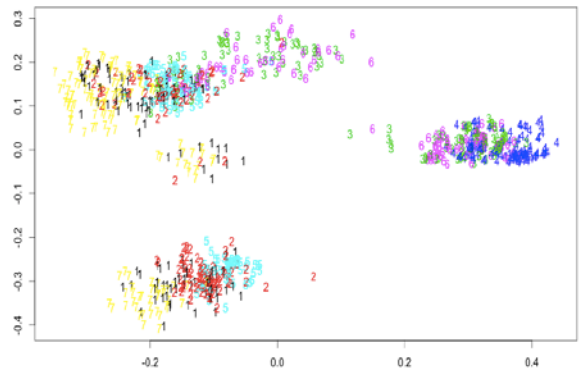
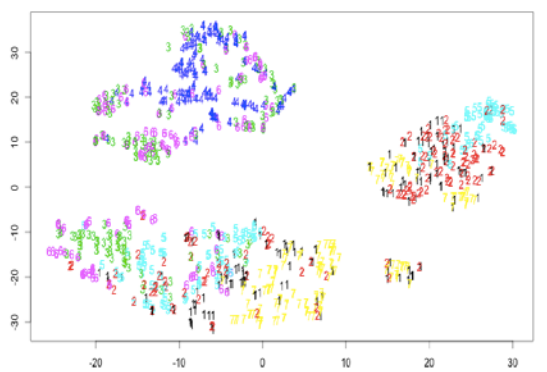
cmdscale() using daisy()	Rtsne() using daisy()
	
cmdscale() using treeClust()	**Rtsne() using treeClust()**
	

Figure 27 shows the plots from taking the long path dimensionality reduction using daisy() function. The plot taking the longer path appears to separate the clusters more obviously, as with the MNIST data. We can see that the t-SNE algorithm produces twists like the MNIST data set (Figures 28 and 30). Figures 29 and 30 display plots taking long path dimensionality reduction using treeClust() function. They do not look as good as in the MNIST data set, but it appears that t-SNE algorithm tries to make close points closer and more distant points farther apart.

Figure 27. Long path of t-SNE of Covertype data using daisy() function

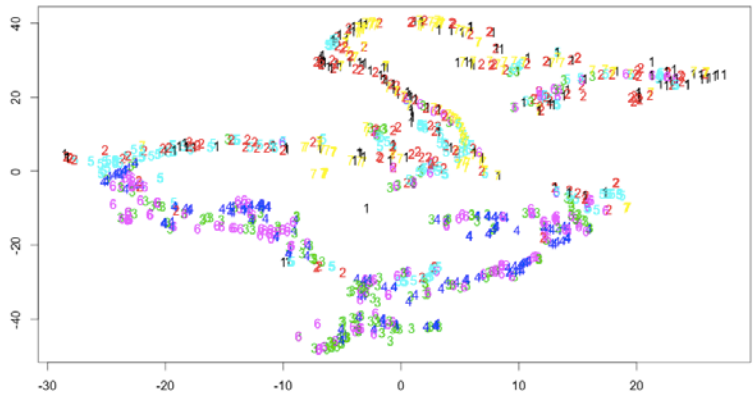
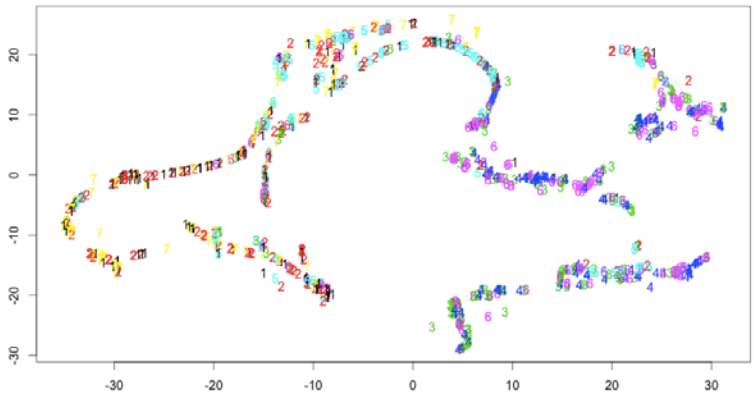
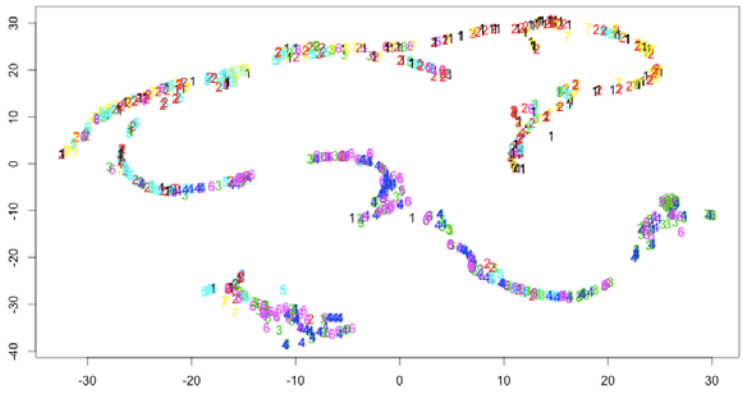
From original dimension to 60 to 2

From original dimension to 60 to 60 to 2

From original dimension to 60 to 50 to 2


Figure 28. Long path of t-SNE of Covertype data using daisy() function

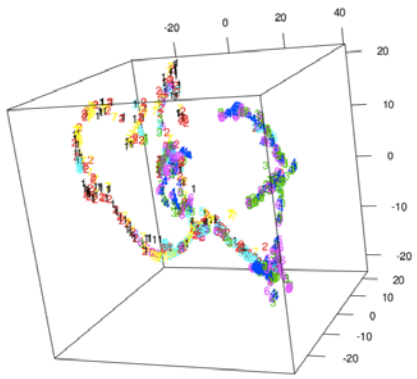
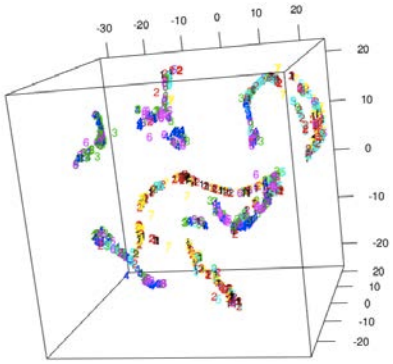
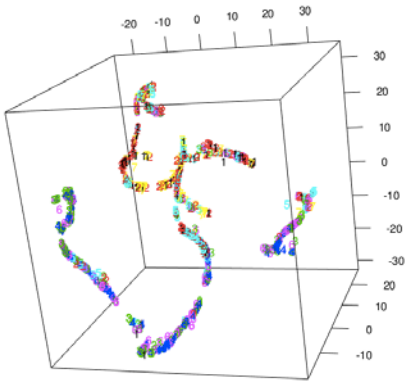
From original dimension to 60 to 3

From original dimension to 60 to 60 to 3

From original dimension to 60 to 50 to 3


Figure 29. Long path of t-SNE of Covertype data using treeClust()

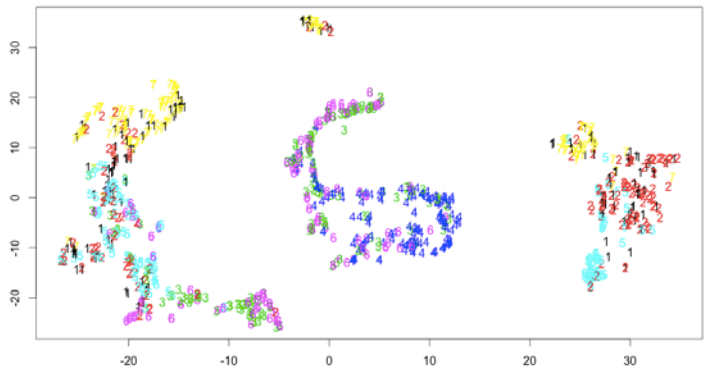
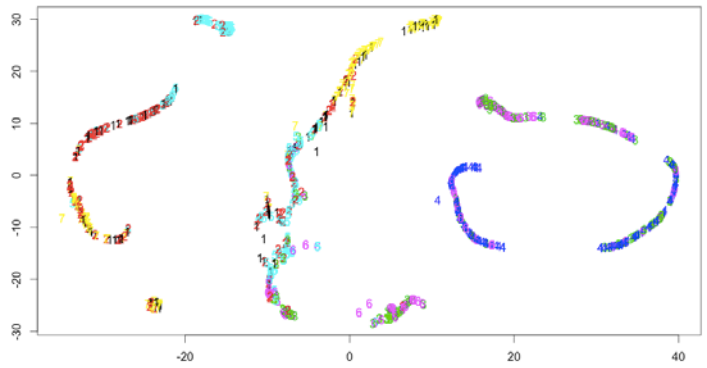
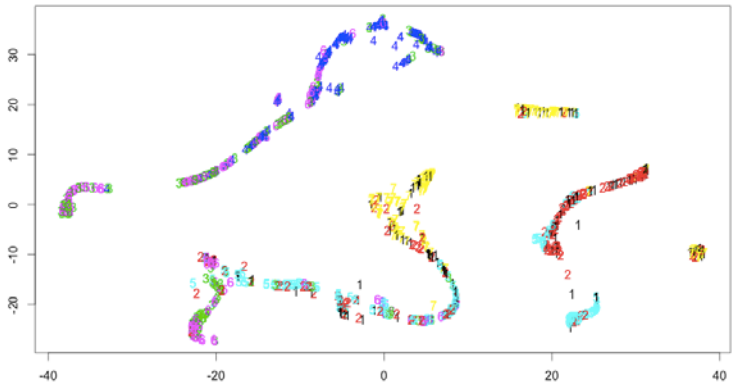
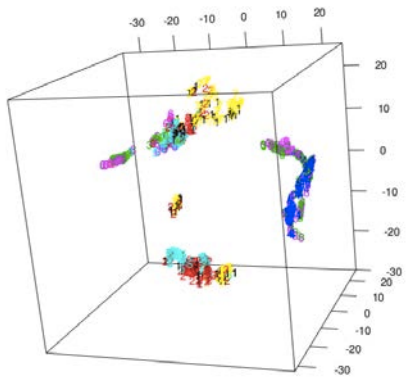
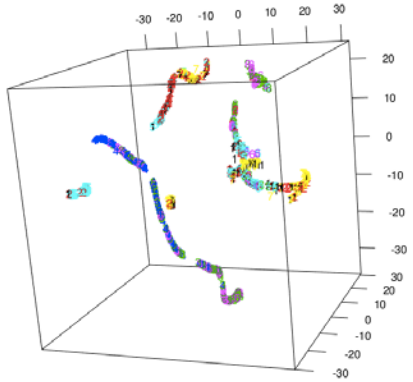
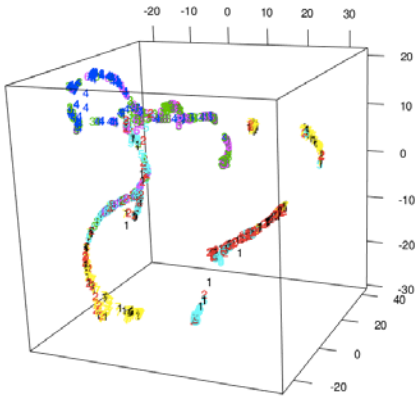
From original dimension to 60 to 2

From original dimension to 60 to 60 to 2

From original dimension to 60 to 50 to 2


Figure 30. Long path of t-SNE of Covertype data using treeClust()

From original dimension to 60 to 3

From original dimension to 60 to 60 to 3

From original dimension to 60 to 50 to 3


THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

Dimensionality reduction is a well-developed area in data analytics. Dimensionality reduction requires a measure of inter-point distance, which requires some thought in the case of mixed or categorical data. How to visualize more purely and clearly is the one of the unsolved problems in analytics, especially for high-dimensional and mixed type data sets. Also the high interest in and demand for big data today makes the visualization more important. We compared the t-SNE algorithm to several multidimensional scaling techniques using both Gower distance and tree distance and explored the dimensionality reduction taking long path using the t-SNE algorithm, which provides an effective way to visualize data sets. We found that the tree distance, which is implemented by the treeClust() function of the treeClust R package, outperforms the Gower distance, which is implemented by the daisy() function of the cluster R package, in our three data sets. We also found that t-SNE algorithm, which is implemented by the Rtsne() function (found in the Rtsne package), outperforms classical multidimensional scaling, which is implemented in the cmdscale() function, in most data sets. So, we conclude that when we use treeClust() and Rtsne() together, we usually get the best picture.

The t-SNE algorithm has some advantages and disadvantages. First, it appears that the t-SNE algorithm visualizes more clearly when we map not just directly from the original, high-dimensional space to two or three dimensions, but via a “long path,” like from “very high” to “high” to “moderate” to “low” dimensions for dimensionality reduction. However, the long path can be computationally difficult and tends to produce twisted, snake-like shapes that can be hard to interpret. Another computational problem arises from duplicates. The t-SNE algorithms cannot operate on data with duplicate entries and some computational effort goes into detecting and removing duplicates. The duplicate problem seems to occur more often when we use the Rtsne() function for relatively small data sets. Fourth, the Rtnse() function’s default setting dimensionality is sixty. In terms

of dimensionality, when we tried the dimensionality reduction from the original dimensions to some number greater than sixty dimensions, it did not perform properly. There were also errors of unknown cause when trying to reduce to fewer than 50 dimensions.

The t-SNE algorithm combined with tree distances gives us a chance to understand high dimensional data sets, and we found some evidence that we can produce more clear and reliable visualizations when we take the long path for dimensionality reduction. We could not find the reason why the Rtsne() function does not work for fewer than fifty dimensions, but it should be considered as future works for more profound dimensionality reduction technologies. Also, when we take long path dimensionality reduction, the algorithm tends to produce twisted shapes. We do not yet know the reason, but if we can figure out that, we can perhaps produce a visualization that is easier to interpret. Reliability Improvements to t-SNE could be very valuable in pursuing these avenues.

LIST OF REFERENCES

Abdi, H. & Williams, L.J. (2010). Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2.

The Apache Software Foundation. (2014). Retrieved from http://hadoop.apache.org/

Burbank, D. (2016). The 5 V's of big data. Retrieved from http://enterprisearchitects.com/the-5v-s-of-big-data/

Buttrey, S. E., & Whitaker, L. R. (2015a). *A scale-independent, noise-resistant dissimilarity for tree-based clustering of mixed data*. Monterey, CA: Naval Postgraduate School.

Buttrey, S. E., & Whitaker, L. R. (2015b). treeClust: An R package for tree-based clustering dissimilarities. *The R Journal*, *7*(2), 227–236.

Buttrey S. E. (2015). Package 'treeClust'. Retrieved from https://cran.r-project.org/web/packages/treeClust/treeClust.pdf

Chen, M., Mao, S., Zhang, Y., & Leung, V. C. M. (2014). *Big data: Related technologies, challenges and future prospects*. New York, NY: Springer.

Cook, J.A., Sutskever, I., Mnih, A., & Hinton, G.E. (2007). Visualizing similarity data with a mixture of maps. In proceedings of the 11th International Conference on Arificial Intelligence and Statistics, *2*, 67–74.

Cramér H. (1999). *Mathematical methods of statistics*, Princeton, NJ: Princeton University Press.

Dave, P. (2013, October 2)., Big data—What is big data—3 Vs of bid data—Volume, velocity and variety. Retrieved from http://blog.sqlauthority.com/2013/10/02/big-data-what-is-big-data-3-vs-of-big-data-volume-velocity-and-variety-day-2-of-21/

Donaldson, J. (2012). tsne: T-distributed Stochastic Neighbor Embedding for R (t-SNE). R package version 0.1-2. Retrieved from https://CRAN.R-project.org/package=tsne

Eynard, D. (2009). Methods for intelligent systems. Retreived from http://davide.eynard.it/teaching/2010_msi/handout-lecture-e1.pdf

Ghodsi, A. (2006). Dimensionality reduction a short tutorial. Retrieved from http://www.math.uwaterloo.ca/~aghodsib/courses/f06stat890/readings/tutorial_stat890.pdf

Gower, J. C. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika, 53*(3/4), 325–338, doi: 10.2307/2333639

Gower J.C. (1971). A general coefficient of similarity and some of its properties, *Biometrics*, *27*, 857–874.

Henderson, P. (1997). Sammon mapping. Retrieved from http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0910/henderson.pdf

Hu, W. C., & Kaabouch, N. (2013). "Big data management, technologies, and applications." IGI Global.

Iliinsky, N. (2012, February 23). Why is data visualization so hot?. Retrieved from http://blog.visual.ly/why-is-data-visualization-so-hot/

Izenman, A J. (2008). *Modern multivariate statistical techniques*. New York, NY: Springer.

Jung, S. (2013). Multidimensional scaling. Retrieved from http://www.stat.pitt.edu/sungkyu/course/2221Fall13/lec8_mds_combined.pdf

Karamizadeh, S., Abdullah, S. M., Manaf, A. A., Zamani, M., & Hooman, A. (2013). An overview of principal component analysis. *Journal of Signal and Information Processing*, *4*, 173–175.

Krijthe, J. (2015, May 26). Package 'Rtsne'. Retrieved from https://cran.r-project.org/web/packages/Rtsne/Rtsne.pdf

Laney, D. (2001). "3D data management: Controlling data volume, velocity, and variety." In *Application Delivery Strategies*, Number 949. Stamford, CT: META Group Inc.

LeCun, Y. (2016). The MNIST database. Retrieved from http://yann.lecun.com/exdb/mnist/

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P.. (1998). "Gradient-based learning applied to document recognition." Proceedings of IEEE, *86*(11), 2278–2324.

Lichman, M. (2013). UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M. & Hornik, K. (2015). Cluster: Cluster analysis basics and extensions. R. package version 2.0.3.

Marr, B. (2015, March 19). Why only one of the 5 Vs of big data really matters. Retrieved from http://www.ibmbigdatahub.com/blog/why-only-one-5-vs-big-data-really-matters

Mearian, L. (2012, Decembr 11). By 2020, there will be 5,200 GB of data for every person on Earth. Retrieved from http://www.computerworld.com/article/2493701/data-center/by-2020--there-will-be-5-200-gb-of-data-for-every-person-on-earth.html

Meyer, B. (2001). Forest cover type prediction. Retrieved from http://homepages.cae.wisc.edu/~ece539/project/f01/meyer.pdf

Olah, C. (2014, October 9). Visualizing MNIST: An exploration of dimensionality reduction. Retrieved from http://colah.github.io/posts/2014-10-Visualizing-MNIST/

Oguntimilehin, A. & Ademola, E.O. (2016). A review of bigh data management, benefits and challenges. Retrieved from http://www.academia.edu/9708014/A_Review_of_Big_Data_Management_Benefits_and_Challenges

Pinal D. (2013, October 2). Big data—What is big data—3 Vs of big data. Retrieved from http://blog.sqlauthority.com/2013/10/02/big-data-what-is-big-data-3-vs-of-big-data-volume-velocity-and-variety-day-2-of-21/

Ray, S. (2015, July 28). Beginners guide to learn dimension reduction techniques. Retrieved from http://www.analyticsvidhya.com/blog/2015/07/dimension-reduction-methods/

R Core Team. (2015). The R Project for statistical computing. Retrieved from http://www.R-project.org.

Rosenberg, A. (2009, February 19). Linear regression with regularization. Retrieved from http://eniac.cs.qc.cuny.edu/andrew/gcml/lecture5.pdf

Rouse, M. (2014, October). Big data analytics. Retrieved from http://searchbusinessanalytics.techtarget.com/definition/big-data-analytics

Sahasrabudhe, N., Machiraju, R., & Zhu, S. C. (2001). A pattern-recognition methodology for enabling digital lighting design. Retrieved from ftp://ftp.cse.ohio-state.edu/pub/tech-report/2001/TR06.ps.gz

SAS. (n.d.). Big data. Retrieved February 25, 2016, from www.sas.com/en_us/insights/big-data/what-is-big-data.html

Van der Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, *15* (2014) 1–21.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *1* (2008) 1–48.

Van der Maaten, L., Postma, E.O., & van den Herik, H.J. (2008, January 11). Dimensionality reduction: A comparative review. Retrieved from http://pages.iai.uni-bonn.de/zimmermann_joerg//dimensionality_reduction_a_comparative_review.pdf

Venables, W.N. & Ripley, B.D. (2002). Modern Applied Statistics with S. Fourth Edition. New York, NY: Springer.

Young, F.W. (1985). Multidimensional scaling. Kotz-Johnson (Ed.) Encyclopedia of Statistical Sciences, Volume 5. Retrieved from http://forrest.psych.unc.edu/teaching/p208a/mds/mds.html

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California